Numerical Linear Algebra
Fundamentals of LA

Zahra Lakdawala

October 2, 2021

Outline

Matrix Multiplications

Definition

- Matrix-vector product $\mathbf{b} = \mathbf{Ax}$

$$b_i = \sum_{j=1}^{n} a_{ij} x_j$$

- All entries belong to $\mathbb{C}$, the field of complex numbers. The space of $m-$vectors is $\mathbb{C}^m$, and the space of $m \times n$ matrices is $\mathbb{C}^{m \times n}$.

Definition

- Matrix-vector product $\mathbf{b} = \mathbf{A}\mathbf{x}$

$$b_i = \sum_{j=1}^{n} a_{ij} x_j$$

- All entries belong to $\mathbb{C}$, the field of complex numbers. The space of $m-$vectors is $\mathbb{C}^m$, and the space of $m \times n$ matrices is $\mathbb{C}^{m \times n}$.
- The map $\mathbf{x} \to \mathbf{A}\mathbf{x}$ is linear, which means for any $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$ and any $\alpha \in \mathbb{C}$

$$\mathbf{A}(\mathbf{x}+\mathbf{y}) = \mathbf{A}\mathbf{x} + \mathbf{A}\mathbf{y}$$

$$\mathbf{A}(\alpha\mathbf{x}) = \alpha\mathbf{A}\mathbf{x}$$

Definition

- Matrix-vector product $\mathbf{b} = \mathbf{A}\mathbf{x}$

$$b_i = \sum_{j=1}^{n} a_{ij} x_j$$

- All entries belong to $\mathbb{C}$, the field of complex numbers. The space of $m-$vectors is $\mathbb{C}^m$, and the space of $m \times n$ matrices is $\mathbb{C}^{m \times n}$.
- The map $\mathbf{x} \to \mathbf{A}\mathbf{x}$ is linear, which means for any $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$ and any $\alpha \in \mathbb{C}$

$$\mathbf{A}(\mathbf{x}+\mathbf{y}) = \mathbf{A}\mathbf{x} + \mathbf{A}\mathbf{y}$$

$$\mathbf{A}(\alpha\mathbf{x}) = \alpha\mathbf{A}\mathbf{x}$$

- In lecture notes, I use **boldface UPPERCASE for matrices, and boldface lowercase letters for vectors.**

Pseudo code for Matrix-Vector Product

This should be straightforward to convert into real computer codes in any programming language.

Pseudo-code for $\mathbf{b} = \mathbf{Ax}$

for $i = 1$ **to** m **do**
    $b(i) = 0$;
    for $j = 1$ **to** n **do**
        $b(i) = b(i) + A(i,j) * x(j)$;
    end for
end for

Linear Combination

- Alternatively, matrix-vector product can be viewed as

$$\mathbf{b} = \mathbf{A}\mathbf{x} = \sum_{j=i}^{n} x_j \mathbf{a}_j$$

i.e **b** is a linear combination of column vectors of **A**

Linear Combination

- Alternatively, matrix-vector product can be viewed as

$$\mathbf{b} = \mathbf{A}\mathbf{x} = \sum_{j=i}^{n} x_j \mathbf{a}_j$$

  i.e **b** is a linear combination of column vectors of **A**

- Two different views of matrix-vector products:
  1. $b_i = \sum_{j=1}^{n} a_{ij} x_j$: **A** acts on **x** to produce **b**; scalar operations
  2. $\mathbf{b} = \sum_{j=i}^{n} x_j \mathbf{a}_j$: **x** acts on **A** to produce **b**; vector operations

Linear Combination

- Alternatively, matrix-vector product can be viewed as

$$\mathbf{b} = \mathbf{A}\mathbf{x} = \sum_{j=i}^{n} x_j \mathbf{a}_j$$

  i.e **b** is a linear combination of column vectors of **A**

- Two different views of matrix-vector products:
  1. $b_i = \sum_{j=1}^{n} a_{ij} x_j$: **A** acts on **x** to produce **b**; scalar operations
  2. $\mathbf{b} = \sum_{j=i}^{n} x_j \mathbf{a}_j$: **x** acts on **A** to produce **b**; vector operations
- If **A** is $m \times n$, **Ax** can be viewed as a mapping from $\mathbb{C}^n$ to $\mathbb{C}^m$

Takeaways

- Space travel: Matrix Vector multiplication takes you from one 'space' to another.
- Two ways to do the same thing
  - We like the product expressed as linear combinations
  - Let's convince ourselves: Algorithmic investigation
- Getting started with Python and numpy package/library.

Matrix-Matrix Multiplication

- If $\mathbf{A}$ is $l \times m$ and $\mathbf{C}$ is $m \times n$, then $\mathbf{B} = \mathbf{AC}$ is $l \times n$, with entries defined by

$$b_{ij} = \sum_{k=1}^{m} a_{ik} c_{kj}.$$

- Written in columns, we have

$$\mathbf{b}_j = \mathbf{A}\mathbf{c}_j = \sum_{k=1}^{m} c_{kj} \mathbf{a}_k.$$

- In other words, each column of $\mathbf{B}$ is a linear combination of the columns of $\mathbf{A}$.

Pseudo-Code for Matrix-Matrix Multiplication

Pseudo-code for $\mathbf{B} = \mathbf{AC}$

```
for i = 1 to l do
    for j = 1 to n do
        B(i,j) = 0;
        for k = 1 to m do
            B(i,j) = B(i,j) + A(i,k) * C(k,j);
        end for
    end for
end for
```

Pseudo-Code for Matrix-Matrix Multiplication

Pseudo-code for $\mathbf{B} = \mathbf{AC}$

```
for i = 1 to l do
    for j = 1 to n do
        B(i,j) = 0;
        for k = 1 to m do
            B(i,j) = B(i,j) + A(i,k) * C(k,j);
        end for
    end for
end for
```

TODO: Write the pseudo code where **B** is expressed as a linear combination of columns of **A**

Rank-1 Matrices

- Full-rank matrices are important
- Another interesting space case is rank-1 matrices
- A matrix $\mathbf{A}$ is rank-1 if it can be written as $\mathbf{A} = \mathbf{u}\mathbf{v}^*$ where $\mathbf{u}$ and $\mathbf{v}$ are non zero vectors
- $\mathbf{u}\mathbf{v}^*$ is called the outer product of the two vectors, as opposed to the inner product $\mathbf{u}^*\mathbf{v}$

Perspective: Vector Space

A useful way in understanding matrix operations is to think in terms of vector spaces

- Vector space spanned by a set of vectors is composed of linear combinations of these vectors
  - It is closed under addition and scalar multiplication
  - $\mathbf{0}$ is always a member of a subspace
  - Space spanned by m-vectors is subspace of $\mathbb{C}^m$
- If S_1 and S_2 are two subspaces, then $S_1 \cap S_2$ is a subspace, so is $S_1 + S_2$, the space of sum of vectors from S_1 and S_2 .
  - Note that $S_1 + S_2$ is different from $S_1 \cup S_2$
- Two subspaces S_1 and S_2 of $\mathbb{C}^m$ are complementary subspaces of each other if $S_1 + S_2 = \mathbb{C}^m$ and $S_1 \cap S_2 = \{\mathbf{0}\}$.
  - In other words, $\dim(S_1\ ) + \dim(S_2\ ) = m$ and $S_1 \cap S_2 = \{\mathbf{0}\}$

Range, Rank and Inverses

Range

Definition

The range of a matrix **A**, written as range(**A**), is the set of vectors that can be expressed as **Ax** for some **x**.

Range

Definition

The range of a matrix $\mathbf{A}$, written as range($\mathbf{A}$), is the set of vectors that can be expressed as $\mathbf{Ax}$ for some $\mathbf{x}$.

Theorem

range($\mathbf{A}$) is the space spanned by the columns of $\mathbf{A}$.

Therefore, the range of $\mathbf{A}$ is also called the column space of $\mathbf{A}$.

Range and Null Space

Definition

The null space of $\mathbf{A} \in \mathbb{C}^{m \times n}$, written as null($\mathbf{A}$), is the set of vectors $\mathbf{x}$ that satisfy $\mathbf{Ax} = \mathbf{0}$.

Entries of $\mathbf{x} \in \mathbf{null}(\mathbf{A})$ give coefficient of $\sum x_i \mathbf{a_i} = \mathbf{0}$. Note: The null space of $\mathbf{A}$ is in general **not** a complimenary subspace of range($\mathbf{A}$).

Rank

Definition

The column rank of a matrix is the dimension of its column space. The row rank is the dimension of the space spanned by its rows.

- Question: Can the column rank and row rank be different?

Rank

Definition

The column rank of a matrix is the dimension of its column space. The row rank is the dimension of the space spanned by its rows.

- Question: Can the column rank and row rank be different?
  - Answer: No
  - Simply referred as the rank of a matrix.

Rank

Definition

The column rank of a matrix is the dimension of its column space. The row rank is the dimension of the space spanned by its rows.

- Question: Can the column rank and row rank be different?
  - Answer: No
  - Simply referred as the rank of a matrix.
- Question: Given $\mathbf{A} \in \mathbb{C}^{m \times n}$, what is $\dim(\text{null}(\mathbf{A})) + \text{rank}(\mathbf{A})$ equal to?

Rank

Definition

The column rank of a matrix is the dimension of its column space. The row rank is the dimension of the space spanned by its rows.

- Question: Can the column rank and row rank be different?
  - Answer: No
  - Simply referred as the rank of a matrix.
- Question: Given $\mathbf{A} \in \mathbb{C}^{m \times n}$, what is $\dim(\text{null}(\mathbf{A})) + \text{rank}(\mathbf{A})$ equal to?
  - Answer: n

Full Rank

Definition

A matrix has *full* rank if it has the maximum possible rank, i.e., $\min\{m, n\}$

Full Rank

Definition

A matrix has *full* rank if it has the maximum possible rank, i.e., $\min\{m, n\}$

Theorem

A matrix $\mathbf{A} \in \mathbb{C}^{m \times n}$ with $m \geq n$ has full rank if and only if it maps no two distinct vectors to the same vector.

In other words, the linear mapping defined by $\mathbf{Ax}$ for $\mathbf{x} \in \mathbb{C}^n$ is one-to-one

Proof

($\Rightarrow$) Column vectors of $\mathbf{A}$ forms a basis of range($\mathbf{A}$), so every $\mathbf{b} \in$ range($\mathbf{A}$) has a unique linear expansion in terms of the columns of $\mathbf{A}$. ($\Leftarrow$) If $\mathbf{A}$ does not have full rank, then its column vectors are linear dependent, so its vectors do not have a unique linear combination

Inverse

Definition

A nonsingular or invertible matrix is a square matrix of full rank.

Inverse

Definition

A nonsingular or invertible matrix is a square matrix of full rank.

Definition

Given a nonsingular matrix $\mathbf{A}$, its inverse is written as $\mathbf{A}^{-1}$, and $\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$

- Note that $(\mathbf{AB})^{-1} = (\mathbf{B}^{-1}\mathbf{A})^{-1}$
- $(\mathbf{A}^{-1})^* = (\mathbf{A}^*)^{-1}$, and we use $\mathbf{A}^{-*}$ as a shorthand for it

Inverse

Theorem

For $\mathbf{A} \in \mathbb{C}^{m \times m}$, the following conditions are equivalent:
(a) $\mathbf{A}$ has an inverse $\mathbf{A}^{-1}$
(b) $rank(\mathbf{A})$ is m
(c) $range(\mathbf{A})$ is $\mathbb{C}^m$
(d) $null(\mathbf{A})$ is $\{0\}$
(e) 0 is not an eigenvalue of $\mathbf{A}$
(f) 0 is not a singular value of $\mathbf{A}$
(g) $det(\mathbf{A}) \neq 0$

Matrix Inverse Times a Vector

- When writing $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$, it means $\mathbf{x}$ is the solution of $\mathbf{Ax} = \mathbf{b}$
- In other words, $\mathbf{A}^{-1}\mathbf{b}$ is a vector of coefficients of the expansion of $\mathbf{b}$ in the basis of columns of $\mathbf{A}$
- Multiplying $\mathbf{b}$ by $\mathbf{A}^{-1}$ is a change of basis operations from $\{\mathbf{a_1}, \mathbf{a_2}, ..., \mathbf{a_m}\}$ to $\{\mathbf{e_1}, \mathbf{e_2}, ..., \mathbf{e_m}\}$
- Multiplying $\mathbf{A}^{-1}\mathbf{b}$ by $\mathbf{A}^{-1}$ is a change of basis operations from $\{\mathbf{e_1}, \mathbf{e_2}, ..., \mathbf{e_m}\}$ to $\{\mathbf{a_1}, \mathbf{a_2}, ..., \mathbf{a_m}\}$

Inner products

Transpose and Adjoint

- Transpose of $\mathbf{A}$, denoted by $\mathbf{A^T}$, is the matrix $\mathbf{B}$ with $b_{ij} = a_{ji}$
- *Adjoint* or *Hermitian conjugate*, denoted by $\mathbf{A}^*$ or $\mathbf{A}^H$, is the matrix $\mathbf{B}$ with $b_{ij} = \bar{a}_{ji}$
- Note that, $(\mathbf{AB})^T = \mathbf{B^T A^T}$ and $(\mathbf{AB})^* = \mathbf{B}^*\mathbf{A}^*$
- A matrix $\mathbf{A}$ is symmetric if $\mathbf{A} = \mathbf{A^T}$ (i.e., $a_{ij} = a_{ji}$). It is *Hermitian* if $\mathbf{A} = \mathbf{A^T}$ (i.e., $a_{ij} = \bar{a}_{ji}$)
- For $\mathbf{A} \in \mathbb{R}^{m \times n}$, null($\mathbf{A}$) and range($\mathbf{A^T}$) are complementary subspaces. In addition, null($\mathbf{A}$) and range($\mathbf{A^T}$) are orthogonal to each other (to be explained later)
- For $\mathbf{A} \in \mathbb{C}^{m \times n}$, null($\mathbf{A}$) and range($\mathbf{A}^*$) are complementary subspaces

Inner Product

- Inner product (dot product) of two column vectors $\mathbf{u}, \mathbf{v} \in \mathbb{C}$ is $\mathbf{u}^*\mathbf{v}$
- In contrast, *outer* product of $\mathbf{u}$ and $\mathbf{v}$ is $\mathbf{u}\mathbf{v}^*$
- Note that *cross* product is different

Inner Product

- Inner product (dot product) of two column vectors $\mathbf{u}, \mathbf{v} \in \mathbb{C}$ is $\mathbf{u}^*\mathbf{v}$
- In contrast, *outer* product of $\mathbf{u}$ and $\mathbf{v}$ is $\mathbf{u}\mathbf{v}^*$
- Note that *cross* product is different

Different ways to see the inner product

1. Vector-vector multiplication: $\mathbf{u}^*\mathbf{v} = \sum_{i=1}^{m} \bar{u}_i v_i$
2. Euclidean length of $\mathbf{u}$ is the square root of the inner product of $\mathbf{u}$ with itself, i.e., $\sqrt{\mathbf{u}^*\mathbf{u}}$
3. Inner product of two unit vectors $\mathbf{u}$ and $\mathbf{v}$ is the cosine of the angle α between $\mathbf{u}$ and $\mathbf{v}$, i.e., $\cos \alpha = \frac{u^* v}{||u||||v||}$

Inner Product is bilinear

Inner product is *bilinear*, in the sense that it is linear in each vertex separately:

- $(u_1 + u_2)^* v = u_1^* v + u_2^* v$
- $u^*(v_1 + v_2) = u^* v_1 + u^* v_2$
- $(\alpha u)^*(\beta v) = \bar{\alpha}\beta u^* v$

Orthogonal Vectors

Definition

A pair of vectors are *orthogonal* if $\mathbf{x}^*\mathbf{y} = 0$.

In other words, the angle between them is 90 degrees

Orthogonal Vectors

Definition

A pair of vectors are *orthogonal* if $\mathbf{x}^*\mathbf{y} = 0$.

In other words, the angle between them is 90 degrees

Definition

Two sets of vectors X and Y are orthogonal if every $\mathbf{x} \in X$ is orthogonal to every $\mathbf{y} \in Y$.

Orthogonal Vectors

Definition

A pair of vectors are *orthogonal* if $\mathbf{x}^*\mathbf{y} = 0$.

In other words, the angle between them is 90 degrees

Definition

Two sets of vectors X and Y are orthogonal if every $\mathbf{x} \in X$ is orthogonal to every $\mathbf{y} \in Y$.

Definition

A set of nonzero vectors S is *orthogonal* if they are pairwise orthogonal. They are *orthonormal* if it is orthogonal and in addition each vector has unit Euclidean length.

Orthogonal Vectors

Theorem

The vectors in an orthogonal set S are linearly independent.

Proof

Prove by contradiction. If a vector can be expressed as linear combination of the other vectors in the set, then it is orthogonal to itself

Question: If the column vectors of an $m \times n$ matrix $\mathbf{A}$ are orthogonal, what is the rank of $\mathbf{A}$?

Orthogonal Vectors

Theorem

The vectors in an orthogonal set S are linearly independent.

Proof

Prove by contradiction. If a vector can be expressed as linear combination of the other vectors in the set, then it is orthogonal to itself

Question: If the column vectors of an $m \times n$ matrix **A** are orthogonal, what is the rank of **A**?
Answer: $n = min\{m, n\}$. In other words, **A** has full rank

Components of Vector

- Given an orthonormal set $\{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_m\}$ forming a basis of $\mathbb{C}^m$, vector $\mathbf{v}$ can be decomposed into orthogonal components as $\mathbf{v} = \sum_{i=1}^{m} (\mathbf{q}_i^* \mathbf{v}) \mathbf{q}_i$

Components of Vector

- Given an orthonormal set $\{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_m\}$ forming a basis of $\mathbb{C}^m$, vector $\mathbf{v}$ can be decomposed into orthogonal components as $\mathbf{v} = \sum_{i=1}^{m} (\mathbf{q}_i^* \mathbf{v}) \mathbf{q}_i$
- Another way to express the condition is $\mathbf{v} = \sum_{i=1}^{m} (\mathbf{q}_i^* \mathbf{q}_i) \mathbf{v}$
- $\mathbf{q}_i \mathbf{q}_i^*$ is an *orthogonal projection matrix*. Note that it is NOT an orthogonal matrix.

Components of Vector

- Given an orthonormal set $\{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_m\}$ forming a basis of $\mathbb{C}^m$, vector $\mathbf{v}$ can be decomposed into orthogonal components as $\mathbf{v} = \sum_{i=1}^{m} (\mathbf{q}_i^* \mathbf{v}) \mathbf{q}_i$
- Another way to express the condition is $\mathbf{v} = \sum_{i=1}^{m} (\mathbf{q}_i^* \mathbf{q}_i) \mathbf{v}$
- $\mathbf{q}_i \mathbf{q}_i^*$ is an *orthogonal projection matrix*. Note that it is NOT an orthogonal matrix.
- More generally, given an orthonormal set $\{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_n\}$ with $n \leq m$, we have

$$\mathbf{v} = \mathbf{r} + \sum_{i=1}^{n} (\mathbf{q}_i^* \mathbf{v}) \mathbf{q}_i = \mathbf{r} + \sum_{i=1}^{n} (\mathbf{q}_i \mathbf{q}_i^*) \mathbf{v} \text{ and } \mathbf{r}^* \mathbf{q}_i = 0,\ 1 \leq i \leq n$$

- Let $\mathbf{Q}$ be composed of column vectors $\{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_n\}$. $\mathbf{Q}\mathbf{Q}^* = \sum_{i=1}^{n} (\mathbf{q}_i \mathbf{q}_i^*)$ is an orthogonal projection matrix.

Unitary matrices

Unitary Matrices

Definition

A matrix is unitary if $\mathbf{Q}^* = \mathbf{Q}^{-1}$,i.e. if $\mathbf{Q}^*\mathbf{Q} = \mathbf{Q}\mathbf{Q}^* = I$

- In the real case, we say the matrix is *orthogonal*. Its column vectors are *orthonormal*.
- In other words, $\mathbf{q}_i^*\mathbf{q}_j = \delta_{ij}$, the *Kronecker delta*

Unitary Matrices

Definition

A matrix is unitary if $\mathbf{Q}^* = \mathbf{Q}^{-1}$,i.e. if $\mathbf{Q}^*\mathbf{Q} = \mathbf{Q}\mathbf{Q}^* = I$

- In the real case, we say the matrix is *orthogonal*. Its column vectors are *orthonormal*.
- In other words, $\mathbf{q}_i^*\mathbf{q}_j = \delta_{ij}$, the *Kronecker delta*

Question: What is the geometric meaning of multiplication by a unitary matrix?

Unitary Matrices

Definition

A matrix is unitary if $\mathbf{Q}^* = \mathbf{Q}^{-1}$,i.e. if $\mathbf{Q}^*\mathbf{Q} = \mathbf{Q}\mathbf{Q}^* = I$

- In the real case, we say the matrix is *orthogonal*. Its column vectors are *orthonormal*.
- In other words, $\mathbf{q}_i^*\mathbf{q}_j = \delta_{ij}$, the *Kronecker delta*

Question: What is the geometric meaning of multiplication by a unitary matrix?
Answer: It preserves angles and Euclidean length. In the real case, multiplication by an orthogonal matrix $\mathbf{Q}$ is a rotation (if $\det(\mathbf{Q}) = 1$) or reflection (if $\det(\mathbf{Q}) = -1$).

Vector Norms

Definition of Norms

- Norm captures "size" of vector or "distance" between vectors
- There are many different measures for "sizes" but a norm must satisfy some requirements:

Definition

A norm is a function $||\cdot||$: $\mathbb{C}^m \rightarrow \mathbb{R}$ that assigns a real-valued length to each vector. It must satisfy the following conditions:

1. $\|\mathbf{x}\| \geq 0$, and $\|\mathbf{x}\| = 0$ only if $\mathbf{x} = 0$
2. $\|\mathbf{x}+\mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$
3. $\|\alpha\mathbf{x}\| = |\alpha|\,\|\mathbf{x}\|$.

- An example is Euclidean length (i.e.$||\mathbf{x}|| = \sqrt{\sum_{i=1}^{m} |x_i|^2}$)

p-norms

- Euclidean length is a special case of p-norms, defined as

$$\|\mathbf{x}\|_p = (\sum_{i=1}^{m} |x_i|^p)^{\frac{1}{p}}$$

for $1 \leq p \leq \infty$

p-norms

- Euclidean length is a special case of p-norms, defined as

$$\|\mathbf{x}\|_p = (\sum_{i=1}^{m} |x_i|^p)^{\frac{1}{p}}$$

  for $1 \leq p \leq \infty$
- Euclidean norm is 2-norm $\|\mathbf{x}\|_2$ (i.e., $p = 2$)

p-norms

- Euclidean length is a special case of p-norms, defined as

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{m} |x_i|^p\right)^{\frac{1}{p}}$$

  for $1 \leq p \leq \infty$
- Euclidean norm is 2-norm $\|\mathbf{x}\|_2$ (i.e., $p = 2$)
- 1-norm: $\|\mathbf{x}\|_1 = \sum_{i=1}^{m} |x_i|$

p-norms

- Euclidean length is a special case of p-norms, defined as

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{m} |x_i|^p\right)^{\frac{1}{p}}$$

  for $1 \leq p \leq \infty$
- Euclidean norm is 2-norm $\|\mathbf{x}\|_2$ (i.e., $p = 2$)
- 1-norm: $\|\mathbf{x}\|_1 = \sum_{i=1}^{m} |x_i|$
- ∞-norm: $\|\mathbf{x}\|_\infty$. What is its value? Answer: $\|\mathbf{x}\|\,|_\infty = \max_{1 \leq i \leq m} |x_i|$
- Why we require $p \geq 1$? What happens if $0 \leq p < 1$?

Weighted p-norms

- A generalization of p-norm is weighted p-norm, which assigns different weights (priorities) to different components.
  - It is anisotropic instead of isotropic
- Algebraically, $\|\mathbf{x}\|_{\mathbf{W}} = \|\mathbf{W}\mathbf{x}\|$, where $\mathbf{W}$ is diagonal matrix with $i-$th diagonal entry $w_i \neq 0$ being weight for ith component
- In other words,

$$\|\mathbf{x}\|_{\mathbf{W}} = (\sum_{i=1}^{m} |w_i x_i|^p)^{\frac{1}{p}}$$

- What happens if we allow $w_i = 0$?

Weighted p-norms

- A generalization of p-norm is weighted p-norm, which assigns different weights (priorities) to different components.
  - It is anisotropic instead of isotropic
- Algebraically, $\|\mathbf{x}\|_{\mathbf{W}} = \|\mathbf{W}\mathbf{x}\|$, where $\mathbf{W}$ is diagonal matrix with $i-$th diagonal entry $w_i \neq 0$ being weight for ith component
- In other words,

$$\|\mathbf{x}\|_{\mathbf{W}} = \left(\sum_{i=1}^{m} |w_i x_i|^p\right)^{\frac{1}{p}}$$

- What happens if we allow $w_i = 0$?
- Can we further generalize it to allow $\mathbf{W}$ being arbitrary matrix?
- No. But we can allow $\mathbf{W}$ to be arbitrary nonsingular matrix.

Matrix Norms

Matrix Norms Induced by Vector Norms

- Viewing $m \times n$ matrix as mn-vectors is not always useful, as operations involving $m \times n$ matrices do not behave this way
- *Induced matrix norms* capture such behavior

Definition

Given vector norms $||\cdot||_{(n)}$ and $||\cdot||_{(m)}$ on domain and range of $\mathbf{A} \in \mathbb{C}^{m \times n}$, respectively, the induced matrix norm $||\mathbf{A}||_{(m,n)}$ is the smallest number $\mathbb{C} \in \mathbb{R}$ for which the following inequality holds for all $\mathbf{x} \in \mathbb{C}^n$:

$$\|\mathbf{Ax}\|_{(m)} \leq C \|x\|_{(n)}.$$

Matrix Norms Induced by Vector Norms

- Viewing $m \times n$ matrix as mn-vectors is not always useful, as operations involving $m \times n$ matrices do not behave this way
- *Induced matrix norms* capture such behavior

Definition

Given vector norms $||\cdot||_{(n)}$ and $||\cdot||_{(m)}$ on domain and range of $\mathbf{A} \in \mathbb{C}^{m \times n}$, respectively, the induced matrix norm $||\mathbf{A}||_{(m,n)}$ is the smallest number $\mathbb{C} \in \mathbb{R}$ for which the following inequality holds for all $\mathbf{x} \in \mathbb{C}^n$:

$$\|\mathbf{A}\mathbf{x}\|_{(m)} \leq C \|x\|_{(n)}.$$

- In other words, it is supremum of ratio $||\mathbf{A}\mathbf{x}||_{(m)}/||\mathbf{x}||_{(n)}$ for all nonzero vectors $\mathbf{x} \in \mathbb{C}^n$
- Maximum factor by which $\mathbf{A}$ can "stretch" $\mathbf{x} \in \mathbb{C}^n$

$$\|\mathbf{A}\|_{(m,n)} = \sup_{\mathbf{x} \in \mathbb{C}^n, \mathbf{x} \neq 0} \|\mathbf{A}\mathbf{x}\|_{(m)} / \|\mathbf{x}\|_{(n)} = \sup_{\mathbf{x} \in \mathbb{C}^n, \|\mathbf{x}\|_{(n)} = 1} \|\mathbf{A}\mathbf{x}\|_{(m)}$$

Matrix Norms Induced by Vector Norms

- Viewing $m \times n$ matrix as mn-vectors is not always useful, as operations involving $m \times n$ matrices do not behave this way
- *Induced matrix norms* capture such behavior

Definition

Given vector norms $||\cdot||_{(n)}$ and $||\cdot||_{(m)}$ on domain and range of $\mathbf{A} \in \mathbb{C}^{m \times n}$, respectively, the induced matrix norm $||\mathbf{A}||_{(m,n)}$ is the smallest number $\mathbb{C} \in \mathbb{R}$ for which the following inequality holds for all $\mathbf{x} \in \mathbb{C}^n$:

$$\|\mathbf{A}\mathbf{x}\|_{(m)} \leq C \|x\|_{(n)}.$$

- In other words, it is supremum of ratio $||\mathbf{A}\mathbf{x}||_{(m)}/||\mathbf{x}||_{(n)}$ for all nonzero vectors $\mathbf{x} \in \mathbb{C}^n$
- Maximum factor by which $\mathbf{A}$ can "stretch" $\mathbf{x} \in \mathbb{C}^n$

$$\|\mathbf{A}\|_{(m,n)} = \sup_{\mathbf{x} \in \mathbb{C}^n, \mathbf{x} \neq 0} \|\mathbf{A}\mathbf{x}\|_{(m)} / \|\mathbf{x}\|_{(n)} = \sup_{\mathbf{x} \in \mathbb{C}^n, \|\mathbf{x}\|_{(n)} = 1} \|\mathbf{A}\mathbf{x}\|_{(m)}$$

- Is vector norm consistent with matrix norm of $m \times 1$-matrix?

1-norm

- By definition

$$||\mathbf{A}||_1 = \sup_{\mathbf{x}\in\mathbb{C}^n, ||\mathbf{x}||_1=1} ||\mathbf{A}\mathbf{x}||_1$$

1-norm

- By definition

$$||\mathbf{A}||_1 = \sup_{\mathbf{x}\in\mathbb{C}^n, ||\mathbf{x}||_1=1} ||\mathbf{A}\mathbf{x}||_1$$

- What is it equal to?

1-norm

- By definition

$$||\mathbf{A}||_1 = \sup_{\mathbf{x}\in\mathbb{C}^n, ||\mathbf{x}||_1=1} ||\mathbf{A}\mathbf{x}||_1$$

- What is it equal to?
  - Maximum of 1-norm of column vectors of $\mathbf{A}$
  - "maximum column sum" of $\mathbf{A}$ is oversimplified in the textbook
- To show it, note that for $\mathbf{x} \in \mathbb{C}^n$ and $||\mathbf{x}||_1 = 1$

$$||\mathbf{A}\mathbf{x}||_1 = ||\sum_{j=1}^{n} x_j \mathbf{a}_j||_1 \leq \max_{1\leq j\leq n} ||a_j||_1 ||\mathbf{x}||_1$$

- Let $k = \arg\max_{1\leq j\leq n} ||\mathbf{a}_j||_1$, then $||\mathbf{A}e_k||_1 = ||\mathbf{a}_k||_1$, so $\max_{1\leq j\leq n} ||a_j||_1$ is tight upper bound

$\infty-$norm

- By definition

$$||\mathbf{A}||_\infty = \sup_{\mathbf{x}\in\mathbb{C}^n, ||\mathbf{x}||_\infty=1} ||\mathbf{A}\mathbf{x}||_\infty$$

- What is $||\mathbf{A}||_\infty$ equal to?

$\infty-$norm

- By definition
$$||\mathbf{A}||_\infty = \sup_{\mathbf{x}\in\mathbb{C}^n, ||\mathbf{x}||_\infty=1} ||\mathbf{A}\mathbf{x}||_\infty$$
- What is $||\mathbf{A}||_\infty$ equal to?
  - Maximum of 1-norm of column vectors of $\mathbf{A}^\mathbf{T}$
- To show it, note that for$\mathbf{x} \in \mathbb{C}^n$ and $||\mathbf{x}||_\infty = 1$
$$||\mathbf{A}\mathbf{x}||_\infty = \max_{1\leq i\leq m}|\mathbf{a}_i^* x| \leq \max_{1\leq i\leq m}||\mathbf{a}_i^*||_1||\mathbf{x}||_\infty$$
  where $\mathbf{a}_i^*$ denotes the $i-$th row vector of $\mathbf{A}$
- Furthermore, $\max_{1\leq i\leq m}||\mathbf{a}_i^*||_1$ is a tight bound.
  - Which vector can we choose to reach the bound?

2-norm

- What is 2-norm of a matrix?

2-norm

- What is 2-norm of a matrix?
- Answer: Its largest singular value.
- We will talk more about singular-value decomposition

2-norm

- What is 2-norm of a matrix?
- Answer: Its largest singular value.
- We will talk more about singular-value decomposition
- What is 2-norm of a diagonal matrix?

Cauchy-Schwarz and Holder Inequalities

- Holder inequality: Let p and q satisfy $1/p + 1/q = 1$ with $1 \leq p,\ q \leq \infty$, then

$$|\mathbf{x}^*\mathbf{y}| \leq ||\mathbf{x}||_p ||\mathbf{y}||_q$$

- Cauchy-Schwarz inequality

$$|\mathbf{x}^*\mathbf{y}| \leq ||\mathbf{x}||_2 ||\mathbf{y}||_2$$

- Cauchy-Schwarz inequality is a special case of Holder inequality

Cauchy-Schwarz and Holder Inequalities

- Holder inequality: Let p and q satisfy $1/p + 1/q = 1$ with $1 \leq p,\ q \leq \infty$, then

$$|\mathbf{x}^*\mathbf{y}| \leq ||\mathbf{x}||_p ||\mathbf{y}||_q$$

- Cauchy-Schwarz inequality

$$|\mathbf{x}^*\mathbf{y}| \leq ||\mathbf{x}||_2 ||\mathbf{y}||_2$$

- Cauchy-Schwarz inequality is a special case of Holder inequality
- Example: What is 2-norm of rank-one matrix? Hint: Use Cauchy-Schwarz inequality.

Bounding Matrix-Matrix Multiplication

- Let $\mathbf{A}$ be an $l \times m$ matrix and $\mathbf{B}$ an $m \times n$ matrix, then for $\mathbf{x} \in \mathbb{C}^n$

$$||\mathbf{AB}||_{(l,n)} \leq ||\mathbf{A}||_{(l,m)} ||\mathbf{B}||_{(m,n)}$$

- To show it, note

$$||\mathbf{ABx}||_{(l)} \leq ||\mathbf{A}||_{(l,m)} ||\mathbf{Bx}||_{(m)} \leq ||\mathbf{A}||_{(l,m)} ||\mathbf{B}||_{(m,n)} ||\mathbf{x}||_{(n)},$$

Bounding Matrix-Matrix Multiplication

- Let $\mathbf{A}$ be an $l \times m$ matrix and $\mathbf{B}$ an $m \times n$ matrix, then for $\mathbf{x} \in \mathbb{C}^n$

$$||\mathbf{AB}||_{(l,n)} \leq ||\mathbf{A}||_{(l,m)} ||\mathbf{B}||_{(m,n)}$$

- To show it, note

$$||\mathbf{ABx}||_{(l)} \leq ||\mathbf{A}||_{(l,m)} ||\mathbf{Bx}||_{(m)} \leq ||\mathbf{A}||_{(l,m)} ||\mathbf{B}||_{(m,n)} ||\mathbf{x}||_{(n)},$$

- In general, this inequality is not an equality
- In particular, $||\mathbf{A}^n|| \leq ||\mathbf{A}||^n$ but $||\mathbf{A}^n|| \neq ||\mathbf{A}||^n$ in general for $n \geq 2$

General Matrix Norms

One can view $m \times n$ matrices as mn-dimensional vectors and obtain *general matrix norms*, which satisfy (for $\mathbf{A}, \mathbf{B} \in \mathbb{C}^{m \times n}$)

1. $||\mathbf{A}|| \geq 0$, and $||\mathbf{A}|| = 0$ only if $\mathbf{A} = 0$
2. $||\mathbf{A} + \mathbf{B}|| \leq ||\mathbf{A}|| + ||\mathbf{B}||$
3. $||\alpha \mathbf{A}|| = |\alpha| ||\mathbf{A}||$

Frobenius Norm

- One useful norm is Frobenius norm (a.k.a. Hilbert-Schmidt norm)

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} = \sqrt{\sum_{j=1}^{n}\|\mathbf{a}_j\|_2^2}$$

  i.e., 2-norm of mn-vector

- Furthermore,

$$\|\mathbf{A}\|_F = \sqrt{tr(\mathbf{A}^\mathbf{T}\mathbf{A})}$$

  where $tr(\mathbf{B})$ denotes trace of B, the sum of its diagonal entries

Frobenius Norm

- One useful norm is Frobenius norm (a.k.a. Hilbert-Schmidt norm)

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} = \sqrt{\sum_{j=1}^{n}\|\mathbf{a}_j\|_2^2}$$

  i.e., 2-norm of mn-vector

- Furthermore,

$$\|\mathbf{A}\|_F = \sqrt{tr(\mathbf{A}^\mathbf{T}\mathbf{A})}$$

  where $tr(\mathbf{B})$ denotes trace of B, the sum of its diagonal entries

- Note that

$$\|\mathbf{AB}\|_F \leq \|\mathbf{A}\|_F \|\mathbf{B}\|_F$$

  because

$$\|\mathbf{AB}\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{m}|\mathbf{a}_i^*\mathbf{b}_j|^2 \leq \sum_{i=1}^{n}\sum_{j=1}^{m}(\|\mathbf{a}_i^*\|_2 \|\mathbf{b}_j\|_2)^2 = \|\mathbf{A}\|_F^2 \|\mathbf{B}\|_F^2$$

Invariance under Unitary Multiplication

Theorem

For any $\mathbf{A} \in \mathbb{C}^{m \times n}$ and unitary $\mathbf{Q} \in \mathbb{C}^{m \times m}$, we have

$$\|\mathbf{QA}\|_2 = \|\mathbf{A}\|_2 \text{ and } \|\mathbf{QA}\|_F = \|A\|_F$$

In other words, 2-norm and Frobenius norms are invariant under unitary multiplication.
Proof for 2-norm: $\|\mathbf{Qy}\|_2 = \|\mathbf{y}\|_2$ for $\mathbf{y} \in \mathbb{C}^m$ and therefore $\|\mathbf{QAx}\|_2 = \|\mathbf{Ax}\|_2$ for $\mathbf{x} \in \mathbb{C}^n$.
It then follows from definition of 2-norm.

SVD

Geometric Observation

- The image of unit sphere under any $m \times n$ matrix is a *hyperellipse*
- Give a unit sphere **S** in $\mathbb{R}^n$, let **AS** denote the shape after transformation
- SVD is

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$$

  where $\mathbf{U} \in \mathbb{C}^{m \times m}$ and $\mathbf{V} \in \mathbb{C}^{n \times n}$ is unitary and $\mathbf{\Sigma} \in \mathbb{R}^{m \times n}$ is diagonal
- Singular values are diagonal entries of $\mathbf{\Sigma}$, correspond to the principal semiaxes, with entries $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$.
- Left singular vectors of **A** are column vectors of **U** and are oriented in the directions of the principal semiaxes of **AS**
- Right singular vectors of **A** are column vectors of **V** and are the preimages of the principal semiaxes of **AS**
- $\mathbf{A}\mathbf{v}_j = \sigma_j \mathbf{u}_j$ for $1 \leq j \leq n$

Two Different Types of SVD

- Full SVD: $\mathbf{U} \in \mathbb{C}^{m \times m}$, $\mathbf{\Sigma} \in \mathbb{R}^{m \times n}$, and $\mathbf{V} \in \mathbb{C}^{n \times n}$ is

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$$

- Reduced SVD: $\hat{\mathbf{U}} \in \mathbb{C}^{m \times n}$, $\hat{\mathbf{\Sigma}} \in \mathbb{R}^{n \times n}$ (assume $m \geq n$)

$$\mathbf{A} = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\mathbf{V}^*$$

- Furthermore, notice that

$$\mathbf{A} = \sum_{i=1}^{min\{m,n\}} \sigma_i \mathbf{u}_i \mathbf{v}_i^*$$

so we can keep only entries of $\mathbf{U}$ and $\mathbf{V}$ corresponding to nonzero σ_i .

Existence of SVD

Theorem

(Existence) Every matrix $\mathbf{A} \in \mathbb{C}^{m \times n}$ has an SVD

Proof: Let $\sigma = \|\mathbf{A}\|_2$. There exists $\mathbf{v}_1 \in \mathbb{C}^n$ with $\|\mathbf{v}_1\|_2 = 1$ and $\|\mathbf{A}\mathbf{v}_1\|_2 = \sigma_1$. Let $\mathbf{U}_1$ and $\mathbf{V}_1$ be unitary matrices whose first columns are $\mathbf{u}_1 = \frac{\mathbf{A}\mathbf{v}_1}{\sigma_1}$ (or any unit-length vector if $\sigma_1 = 0$) and $\mathbf{v}_1$, respectively. Note that

$$\mathbf{U}_1^* \mathbf{A} \mathbf{V}_1 = \mathbf{S} = \begin{bmatrix} \sigma_1 & \omega^* \\ \mathbf{0} & \mathbf{B} \end{bmatrix}$$

Furthermore, $\omega = 0$ because $\|\mathbf{S}\|_2 = \sigma_1$, and

$$\left\| \begin{bmatrix} \sigma_1 & \omega^* \\ \mathbf{0} & \mathbf{B} \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \omega \end{bmatrix} \right\| \geq \sigma_1^2 + \omega^* \omega = \sqrt{\sigma_1^2 + \omega^* \omega} \left\| \begin{bmatrix} \sigma_1 \\ \omega \end{bmatrix} \right\|_2,$$

implying that $\omega_1 \geq \sqrt{\sigma_1^2 + \omega^* \omega}$ and $\omega = 0$

Existence of SVD Cont'd

We then prove by induction using (1). If $m = 1$ or $n = 1$, then $\mathbf{B}$ is empty and we have $\mathbf{A} = \mathbf{U}_1 \mathbf{S} \mathbf{V}_1^*$. Otherwise, suppose $\mathbf{B} = \mathbf{U}_2 \mathbf{\Sigma}_2 \mathbf{V}_2^*$, and then

$$\mathbf{A} = \underbrace{\mathbf{U}_1 \begin{bmatrix} 1 & \mathbf{0}^* \\ \mathbf{0} & \mathbf{U}_2 \end{bmatrix}}_{\mathbf{U}} \underbrace{\begin{bmatrix} \sigma_1 & \mathbf{0}^* \\ \mathbf{0} & \mathbf{\Sigma}_2 \end{bmatrix}}_{\mathbf{\Sigma}}, \underbrace{\begin{bmatrix} 1 & \mathbf{0}^* \\ \mathbf{0} & \mathbf{V}_2^* \end{bmatrix} \mathbf{V}_1^*}_{\mathbf{V}^*}$$

where $\mathbf{U}$ and $\mathbf{V}$ are unitary.

Uniquesness of SVD

Theorem

(Uniqueness) The singular values $\{\sigma_j\}$ are uniquely determined. If **A** is square and the σ_j are distinct, the left and right singular vectors are uniquely determined **up to complex signs** (i.e., complex scalar factors of absolute value 1).

Geometric argument: If the lengths of semiaxes of a hyperellipse are distinct, then the semiaxes themselves are determined by the geometry up to signs.

Uniqueness of SVD Cont'd

Algebraic argument: Based on 2-norm and prove by induction. Consider the case where the σ_j are distinct. The 2-norm is unique, so is σ_1 . If $\mathbf{v}_1$ is not unique up to sign, then the orthonormal bases of these vectors are right singular vectors of $\mathbf{A}$, implying that σ_1 is not a simple singular value.
Once σ_1 , $\mathbf{u}_1$, and $\mathbf{v}_1$ are determined, the remainder of SVD is determined by the space orthogonal to $\mathbf{v}_1$. Because $\mathbf{v}_1$ is unique up to sign, the orthogonal subspace is uniquely defined. Then prove by induction.

Uniqueness of SVD Cont'd

Algebraic argument: Based on 2-norm and prove by induction. Consider the case where the σ_j are distinct. The 2-norm is unique, so is σ_1 . If $\mathbf{v}_1$ is not unique up to sign, then the orthonormal bases of these vectors are right singular vectors of $\mathbf{A}$, implying that σ_1 is not a simple singular value.
Once σ_1 , $\mathbf{u}_1$, and $\mathbf{v}_1$ are determined, the remainder of SVD is determined by the space orthogonal to $\mathbf{v}_1$. Because $\mathbf{v}_1$ is unique up to sign, the orthogonal subspace is uniquely defined. Then prove by induction.

- Question: What if we change the sign of a singular vector?

Uniqueness of SVD Cont'd

Algebraic argument: Based on 2-norm and prove by induction. Consider the case where the σ_j are distinct. The 2-norm is unique, so is σ_1 . If $\mathbf{v}_1$ is not unique up to sign, then the orthonormal bases of these vectors are right singular vectors of $\mathbf{A}$, implying that σ_1 is not a simple singular value.
Once σ_1 , $\mathbf{u}_1$, and $\mathbf{v}_1$ are determined, the remainder of SVD is determined by the space orthogonal to $\mathbf{v}_1$. Because $\mathbf{v}_1$ is unique up to sign, the orthogonal subspace is uniquely defined. Then prove by induction.

- Question: What if we change the sign of a singular vector?
- Question: What if σ_i is not distinct?

SVD vs Eigenvalue Decomposition

- Eigenvalue decomposition of nondefective matrix $\mathbf{A}$ is $\mathbf{A} = \mathbf{X}\mathbf{\Lambda}\mathbf{X}^{-1}$

Differences

- Not every matrix has eigenvalue decomposition, but every matrix has singular value decomposition
- Eigenvalues may not always be real numbers, but singular values are always non-negative real numbers
- Eigenvectors are not always orthogonal to each other (orthogonal for symmetric matrices), but left (or right) singular vectors are orthogonal to each other

SVD vs Eigenvalue Decomposition

Similarities

- Singular values of $\mathbf{A}$ are square roots of eigenvalues of $\mathbf{AA}^*$ and $\mathbf{A}^*\mathbf{A}$, and their eigenvectors are left and right singular vectors, respectively
- Singular values of hermitian matrices are absolute values of eigenvalues, and eigenvectors are singular vectors (up to complex signs)
- This relationship can be used to compute singular values by hand

Matrix Properties via SVD

- Let r be number of nonzero singular values of $\mathbf{A} \in \mathbb{C}^{m \times n}$
  - rank($\mathbf{A}$) is r
  - range($\mathbf{A}$) $=< \mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_r >$
  - null($\mathbf{A}$) $=< \mathbf{u}_{r+1}, \mathbf{u}_{r+2}, ..., \mathbf{u}_n >$
- 2-norm and Frobenius norm
  - $\|\mathbf{A}\|_2 = \sigma_1$ and $\|\mathbf{A}\|_F = \sqrt{\sum_i \sigma_i^2}$
- Determinant of matrix
  - For $\mathbf{A} \in \mathbb{C}^{m \times m}$, $|det(\mathbf{A})| = \Pi_{i=1}^{m} \sigma_i$

Matrix Properties via SVD

- Let r be number of nonzero singular values of $\mathbf{A} \in \mathbb{C}^{m \times n}$
  - rank($\mathbf{A}$) is r
  - range($\mathbf{A}$) $=< \mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_r >$
  - null($\mathbf{A}$) $=< \mathbf{u}_{r+1}, \mathbf{u}_{r+2}, ..., \mathbf{u}_n >$
- 2-norm and Frobenius norm
  - $\|\mathbf{A}\|_2 = \sigma_1$ and $\|\mathbf{A}\|_F = \sqrt{\sum_i \sigma_i^2}$
- Determinant of matrix
  - For $\mathbf{A} \in \mathbb{C}^{m \times m}$, $|det(\mathbf{A})| = \Pi_{i=1}^{m} \sigma_i$
- However, SVD may not be the most efficient way in solving problems
- Algorithms for SVD are similar to those for eigenvalue decomposition and we will discuss them later in the semester

Numerical Linear Algebra
QR and Least Squares

Zahra Lakdawala

October 7, 2021

Outline

Projectors

Projectors

- A projector satisfies $\mathbf{P}^2 = \mathbf{P}$. They are also said to be *idempotent*.
  - Orthogonal projector
  - Oblique projector
- Example

$$\begin{bmatrix} 0 & 0 \\ \alpha & 1 \end{bmatrix}$$

  - is an oblique projector if $\alpha \neq 0$,
  - is orthogonal projector if $\alpha = 0$.

Complementary Projectors

- Complementary projectors: $\mathbf{P}$ vs. $\mathbf{I}-\mathbf{P}$.
- What space does $\mathbf{I}-\mathbf{P}$ project?

Complementary Projectors

- Complementary projectors: $\mathbf{P}$ vs. $\mathbf{I}-\mathbf{P}$.
- What space does $\mathbf{I}-\mathbf{P}$ project?
  - Answer: null($\mathbf{P}$)
  - range($\mathbf{I}-\mathbf{P}$) $\supseteq$ null($\mathbf{P}$) because $\mathbf{Pv}=0 \Rightarrow (\mathbf{I}-\mathbf{P})\mathbf{v}=\mathbf{v}$.
  - range($\mathbf{I}-\mathbf{P}$) $\subseteq$ null($\mathbf{P}$) because for any $\mathbf{v}$

$$(\mathbf{I}-\mathbf{P})\mathbf{v}=\mathbf{v}-\mathbf{Pv}\in \text{null}(\mathbf{P}).$$

- A projector separates $\mathbb{C}^m$ into two complementary subspace: range space and null space (i.e., range($\mathbf{P}$) + null($\mathbf{P}$) $=\mathbb{C}^m$ and range($\mathbf{P}$) $\cap$ null($\mathbf{P}$) $=0$ for projector $\mathbf{P}\in\mathbb{C}^{m\times m}$)
- It projects onto range space along null space
  - In other words, $\mathbf{x}=\mathbf{Px}+\mathbf{r}$, where $\mathbf{r}\in \textit{null}(\mathbf{P})$
- Question: Are range space and null space of projector orthogonal to each other?

Orthogonal Projector

- An orthogonal projector is one that projects onto a subspace S_1 along a space S_2, where S_1 and S_2 are orthogonal.

Theorem

A projector $\mathbf{P}$ is orthogonal if and only if $\mathbf{P} = \mathbf{P}^*$.

Proof

"If" direction: If $\mathbf{P} = \mathbf{P}^*$, then$(\mathbf{Px})^*(\mathbf{I}-\mathbf{P})\mathbf{y} = \mathbf{x}^*(\mathbf{P}-\mathbf{P}^2)\mathbf{y}$. "Only if" direction: Use SVD. Suppose $\mathbf{P}$ projects onto S_1 along S_2 where $S_1 \perp S_2$, and S_1 has dimension n. Let $q_1, \cdot, q_n$ be orthonormal basis of S_1 and $q_{n+1}, \cdot, q_m$ be a basis for S_2. Let $\mathbf{Q}$ be unitary matrix whose jth column is q_j , and we have $\mathbf{PQ} = (\mathbf{q}_1, \mathbf{q}_2, \cdot \mathbf{q}_n, 0, \cdot, 0)$, so $\mathbf{Q}^*\mathbf{PQ} = diag(1,1,\cdot,1,0,\cdot) = \Sigma$, and $\mathbf{P} = \mathbf{Q}\Sigma\mathbf{Q}^*$.

Question: Are orthogonal projectors orthogonal matrices?

Basis of Projections

- Projection with orthonormal basis
  - Given any matrix $\hat{\mathbf{Q}} \in \mathbb{C}^{m \times n}$ whose columns are orthonormal, then $\mathbf{P} = \hat{\mathbf{Q}}\hat{\mathbf{Q}}^*$ is orthogonal projector, so is $\mathbf{I} - \mathbf{P}$
  - We write $\mathbf{I} - \mathbf{P}$ as $\mathbf{P}_\perp$
  - In particular, if $\hat{\mathbf{Q}} = \mathbf{q}$, we write $\mathbf{P_q} = \mathbf{q}\mathbf{q}^*$ and $\mathbf{P}_{\perp\mathbf{q}} = \mathbf{I} - \mathbf{P_q}$
  - For arbitrary vector $\mathbf{a}$, we write $\mathbf{P_a} = \frac{\mathbf{a}\mathbf{a}^*}{\mathbf{a}^*\mathbf{a}}$ and $\mathbf{P}_{\perp\mathbf{a}} = \mathbf{I} - \mathbf{P_a}$

Basis of Projections

- Projection with arbitrary basis
  - Given any matrix $\mathbf{A} \in \mathbb{C}^{m \times n}$ that has full rank $m \geq n$

$$\mathbf{P} = \mathbf{A}(\mathbf{A}^*\mathbf{A})^{-1}\mathbf{A}^*$$

    is an orthogonal projection
  - What does $\mathbf{P}$ project onto?
    - range($\mathbf{A}$)
  - $(\mathbf{A}^*\mathbf{A})^{-1}\mathbf{A}^*$ is called the pseudo inverse of $\mathbf{A}$, denoted as $\mathbf{A}^+$

QR Factorization

Motivation

- Question: Given a linear system $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A} \in \mathbb{C}^{m \times n} (m \geq n)$ has full rank, how to solve the linear system?
- Answer: One possible solution is to use SVD. How?

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^*, \text{so } \mathbf{x} = \mathbf{V}\Sigma^{-1}\mathbf{U}^*\mathbf{b}$$

Motivation

- Question: Given a linear system $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A} \in \mathbb{C}^{m \times n} (m \geq n)$ has full rank, how to solve the linear system?
- Answer: One possible solution is to use SVD. How?

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^*, \text{so } \mathbf{x} = \mathbf{V}\Sigma^{-1}\mathbf{U}^*\mathbf{b}$$

Another solution is to use QR factorization, which decompose $\mathbf{A}$ into product of two simple matrices $\mathbf{Q}$ and $\mathbf{R}$ where columns of $\mathbf{Q}$ are orthonormal and $\mathbf{R}$ is upper triangular.

Two Different Versions of QR

- Full QR factorization: $\mathbf{A} \in \mathbb{C}^{m \times n} (m \geq n)$

$$\mathbf{A} = \mathbf{QR}$$

  where $\mathbf{Q} \in \mathbb{C}^{m \times m}$ is unitary and $\mathbf{R} \in \mathbb{C}^{m \times n}$ is upper trianglular
- Reduced QR factorization: $\mathbf{A} \in \mathbb{C}^{m \times n} (m \geq n)$

$$\mathbf{A} = \hat{\mathbf{Q}}\hat{\mathbf{R}}$$

  where $\hat{\mathbf{Q}} \in \mathbb{C}^{m \times n}$ contains orthonormal vectors and $\hat{\mathbf{R}} \in \mathbb{C}^{n \times n}$ is upper triangular
- What space do $\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_j, j \leq n$ span?

Two Different Versions of QR

- Full QR factorization: $\mathbf{A} \in \mathbb{C}^{m \times n} (m \geq n)$

$$\mathbf{A} = \mathbf{QR}$$

  where $\mathbf{Q} \in \mathbb{C}^{m \times m}$ is unitary and $\mathbf{R} \in \mathbb{C}^{m \times n}$ is upper trianglular
- Reduced QR factorization: $\mathbf{A} \in \mathbb{C}^{m \times n} (m \geq n)$

$$\mathbf{A} = \hat{\mathbf{Q}}\hat{\mathbf{R}}$$

  where $\hat{\mathbf{Q}} \in \mathbb{C}^{m \times n}$ contains orthonormal vectors and $\hat{\mathbf{R}} \in \mathbb{C}^{n \times n}$ is upper triangular
- What space do $\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_j, j \leq n$ span?
  - Answer: For full rank $\mathbf{A}$, first j column vectors of $\mathbf{A}$, i.e. $\langle \mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_j \rangle = \langle \mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_j \rangle$

Gram-Schmidt Orthogonalization

- A method to construct QR factorization is to orthogonalize the column vectors of $\mathbf{A}$:
- Basic idea:
  - Take first column $\mathbf{a}_1$ and normalize it to obtain vector $\mathbf{q}_1$;
  - Take second column $\mathbf{a}_2$, subtract its orthogonal projection to $\mathbf{q}_1$, and normalize to obtain $\mathbf{q}_2$;
  - $\cdots$
  - Take j-th column of $\mathbf{a}_j$, subtract its orthogonal projection to $\mathbf{q}_1, \cdots, \mathbf{q}_{j-1}$ and normalize to obtain $\mathbf{q}_j$

$$\mathbf{v}_j = \mathbf{a}_j - \sum_{i=1}^{j-1} \mathbf{q}_i^* \mathbf{a}_j \mathbf{q}_i, \quad \mathbf{q}_j = \mathbf{v}_j / \|\mathbf{v}_j\|$$

- This idea is called Gram-Schmidt orthogonalization.

Gram Schmidt Projections

- Orthogonal vectors produced by Gram-Schmidt can be written in terms of projectors

$$\mathbf{q}_j = \frac{\mathbf{P}_j \mathbf{a}_j}{\|\mathbf{P}_j \mathbf{a}_j\|}$$

where

$$\mathbf{P}_j = \mathbf{I} - \hat{\mathbf{Q}}_{j-1}\hat{\mathbf{Q}}_{j-1}^* \text{ with } \hat{\mathbf{Q}}_{j-1} = [\mathbf{q}_1 \quad \mathbf{q}_2 \quad \cdots \mathbf{q}_{j-1}]$$

- $\mathbf{P}_j$ projects orthogonally onto space orthogonal to $\langle \mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_{j-1} \rangle$ and rank of $\mathbf{P}_j$ is $m-(j-1)$

Algorithm of Gram Schmidt Orthogonalization

Classical Gram-Schmidt method

for $j = 1$ **to** n
  $\mathbf{v}_j = \mathbf{a}_j$;
  for $i = 1$ **to** $j - 1$
    $r_{ij} = \mathbf{q}_i^* \mathbf{a}_j$
    $\mathbf{v}_j = \mathbf{v}_j - r_{ij}\mathbf{q}_i$
  $r_{jj} = \|\mathbf{v}_j\|_2$
  $\mathbf{q}_j = \frac{\mathbf{v}_j}{r_{jj}}$

- Classical Gram-Schmidt (CGS) is unstable, which means that its solution is sensitive to perturbation

Existence of QR

Theorem

Every $\mathbf{A} \in \mathbb{C}^{m \times n} (m \geq n)$ has full QR factorization, hence also a reduced QR factorization.

Key idea of proof:

- If $\mathbf{A}$ has full rank, Gram-Schmidt algorithm provides a proof itself for having reduced QR.
- If $\mathbf{A}$ does not have full rank, at some step $\mathbf{v}_j = 0$. We can set $\mathbf{q}_j$ to be a vector orthogonal to $\mathbf{q}_i, i < j$.
- To construct full QR from reduced QR, just continue Gram-Schmidt an additional $m - n$ steps.

Uniqueness of QR

Theorem

Every $\mathbf{A} \in \mathbb{C}^{m \times n} (m \geq n)$ has full rank has a unique reduced QR factorization $\mathbf{A} = \hat{\mathbf{Q}}\hat{\mathbf{R}}$ with $r_{jj} > 0$.

Proof is provided by Gram-Schmidt iteration itself. If the signs of r_{jj} are determined, then r_{ij} and $\mathbf{q}_j$ are determined.

Uniqueness of QR

Theorem

Every $\mathbf{A} \in \mathbb{C}^{m \times n} (m \geq n)$ has full rank has a unique reduced QR factorization $\mathbf{A} = \hat{\mathbf{Q}}\hat{\mathbf{R}}$ with $r_{jj} > 0$.

Proof is provided by Gram-Schmidt iteration itself. If the signs of r_{jj} are determined, then r_{ij} and $\mathbf{q}_j$ are determined.
Question: Why do we require $r_{jj} > 0$

Uniqueness of QR

Theorem

Every $\mathbf{A} \in \mathbb{C}^{m \times n} (m \geq n)$ has full rank has a unique reduced QR factorization $\mathbf{A} = \hat{\mathbf{Q}}\hat{\mathbf{R}}$ with $r_{jj} > 0$.

Proof is provided by Gram-Schmidt iteration itself. If the signs of r_{jj} are determined, then r_{ij} and $\mathbf{q}_j$ are determined.
Question: Why do we require $r_{jj} > 0$
Question: Is full QR factorization unique?

Uniqueness of QR

Theorem

Every $\mathbf{A} \in \mathbb{C}^{m \times n} (m \geq n)$ has full rank has a unique reduced QR factorization $\mathbf{A} = \hat{\mathbf{Q}}\hat{\mathbf{R}}$ with $r_{jj} > 0$.

Proof is provided by Gram-Schmidt iteration itself. If the signs of r_{jj} are determined, then r_{ij} and $\mathbf{q}_j$ are determined.
Question: Why do we require $r_{jj} > 0$
Question: Is full QR factorization unique?
Question: What if A does not have full rank?

Gram-Schmidt Orthogonalization

Gram-Schmidt Orthogonalization

- A method to construct QR factorization is to orthogonalize the column vectors of $\mathbf{A}$:
- Basic idea:
  - Take first column $\mathbf{a}_1$ and normalize it to obtain vector $\mathbf{q}_1$;
  - Take second column $\mathbf{a}_2$, subtract its orthogonal projection to $\mathbf{q}_1$, and normalize to obtain $\mathbf{q}_2$;
  - $\cdots$
  - Take j-th column of $\mathbf{a}_j$, subtract its orthogonal projection to $\mathbf{q}_1, \cdots, \mathbf{q}_{j-1}$ and normalize to obtain $\mathbf{q}_j$

$$\mathbf{v}_j = \mathbf{a}_j - \sum_{i=1}^{j-1} \mathbf{q}_i^* \mathbf{a}_j \mathbf{q}_i, \quad \mathbf{q}_j = \mathbf{v}_j / \|\mathbf{v}_j\|$$

- This idea is called Gram-Schmidt orthogonalization.

Gram Schmidt Projections

- Orthogonal vectors produced by Gram-Schmidt can be written in terms of projectors

$$\mathbf{q}_j = \frac{\mathbf{P}_j \mathbf{a}_j}{\|\mathbf{P}_j \mathbf{a}_j\|}$$

where

$$\mathbf{P}_j = \mathbf{I} - \hat{\mathbf{Q}}_{j-1}\hat{\mathbf{Q}}_{j-1}^* \text{ with } \hat{\mathbf{Q}}_{j-1} = [\mathbf{q}_1 \quad \mathbf{q}_2 \quad \cdots \mathbf{q}_{j-1}]$$

- $\mathbf{P}_j$ projects orthogonally onto space orthogonal to $\langle \mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_{j-1} \rangle$ and rank of $\mathbf{P}_j$ is $m-(j-1)$

Algorithm of Gram Schmidt Orthogonalization

Classical Gram-Schmidt method

for $j = 1$ **to** n
  $\mathbf{v}_j = \mathbf{a}_j$;
  for $i = 1$ **to** $j - 1$
    $r_{ij} = \mathbf{q}_i^* \mathbf{a}_j$
    $\mathbf{v}_j = \mathbf{v}_j - r_{ij}\mathbf{q}_i$
  $r_{jj} = \|\mathbf{v}_j\|_2$
  $\mathbf{q}_j = \frac{\mathbf{v}_j}{r_{jj}}$

- Classical Gram-Schmidt (CGS) is unstable, which means that its solution is sensitive to perturbation

Existence of QR

Theorem

Every $\mathbf{A} \in \mathbb{C}^{m \times n} (m \geq n)$ has full QR factorization, hence also a reduced QR factorization.

Key idea of proof:

- If $\mathbf{A}$ has full rank, Gram-Schmidt algorithm provides a proof itself for having reduced QR.
- If $\mathbf{A}$ does not have full rank, at some step $\mathbf{v}_j = 0$. We can set $\mathbf{q}_j$ to be a vector orthogonal to $\mathbf{q}_i, i < j$.
- To construct full QR from reduced QR, just continue Gram-Schmidt an additional $m - n$ steps.

Uniqueness of QR

Theorem

Every $\mathbf{A} \in \mathbb{C}^{m \times n} (m \geq n)$ has full rank has a unique reduced QR factorization $\mathbf{A} = \hat{\mathbf{Q}}\hat{\mathbf{R}}$ with $r_{jj} > 0$.

Proof is provided by Gram-Schmidt iteration itself. If the signs of r_{jj} are determined, then r_{ij} and $\mathbf{q}_j$ are determined.

Uniqueness of QR

Theorem

Every $\mathbf{A} \in \mathbb{C}^{m \times n} (m \geq n)$ has full rank has a unique reduced QR factorization $\mathbf{A} = \hat{\mathbf{Q}}\hat{\mathbf{R}}$ with $r_{jj} > 0$.

Proof is provided by Gram-Schmidt iteration itself. If the signs of r_{jj} are determined, then r_{ij} and $\mathbf{q}_j$ are determined.
Question: Why do we require $r_{jj} > 0$

Uniqueness of QR

Theorem

Every $\mathbf{A} \in \mathbb{C}^{m \times n} (m \geq n)$ has full rank has a unique reduced QR factorization $\mathbf{A} = \hat{\mathbf{Q}}\hat{\mathbf{R}}$ with $r_{jj} > 0$.

Proof is provided by Gram-Schmidt iteration itself. If the signs of r_{jj} are determined, then r_{ij} and $\mathbf{q}_j$ are determined.
Question: Why do we require $r_{jj} > 0$
Question: Is full QR factorization unique?

Uniqueness of QR

Theorem

Every $\mathbf{A} \in \mathbb{C}^{m \times n} (m \geq n)$ has full rank has a unique reduced QR factorization $\mathbf{A} = \hat{\mathbf{Q}}\hat{\mathbf{R}}$ with $r_{jj} > 0$.

Proof is provided by Gram-Schmidt iteration itself. If the signs of r_{jj} are determined, then r_{ij} and $\mathbf{q}_j$ are determined.
Question: Why do we require $r_{jj} > 0$
Question: Is full QR factorization unique?
Question: What if A does not have full rank?

Alternative view to Gram-Schmidt Projection

- Orthogonal vectors produced by Gram-Schmidt can be written in terms of projectors

$$\mathbf{q}_j = \frac{\mathbf{P}_j \mathbf{a}_j}{\|\mathbf{P}_j \mathbf{a}_j\|},$$

  where $\mathbf{P}_j = \mathbf{I} - \hat{\mathbf{Q}}_{j-1}\hat{\mathbf{Q}}_{j-1}^*$ with $\hat{\mathbf{Q}}_{j-1} = [\mathbf{q}_1\ \mathbf{q}_2 \ldots \mathbf{q}_{j-1}]$
- We may view $\mathbf{P}_j$ as product of a sequence of projections

$$\mathbf{P}_j = \mathbf{P}_{\perp q_{j-1}} \mathbf{P}_{\perp q_{j-2}} \ldots \mathbf{P}_{\perp q_1}$$

  where $\mathbf{P}_{\perp q} = \mathbf{I} - \mathbf{q}\mathbf{q}^*$
- Instead of computing $\mathbf{v}_j = \mathbf{P}_j \mathbf{a}_i$, one could compute $\mathbf{v}_j = \mathbf{P}_{\perp q_{j-1}} \mathbf{P}_{\perp q_{j-2}} \ldots \mathbf{P}_{\perp q_1} a_j$ instead, resulting in modified Gram-Schmidt algorithm

Modified Gram-Schmidt Orthogonalization

Classical Gram-Schmidt method:

for $j = 1$ **to** n

$\mathbf{v}_j = \mathbf{a}_j$;

for $i = 1$ **to** $j-1$

$r_{ij} = \mathbf{q}_i^* \mathbf{a}_j$

$\mathbf{v}_j = \mathbf{v}_j - r_{ij}\mathbf{q}_i$

$r_{jj} = \|\mathbf{v}_j\|_2$

$\mathbf{q}_j = \frac{\mathbf{v}_j}{r_{jj}}$

Modified Gram-Schmidt method:

for $j = 1$ to n

$\mathbf{v}_j = \mathbf{a}_j$

for $i = 1$ to n

$r_{ii} = \|\mathbf{v}_i\|_2$

$\mathbf{q}_i = \mathbf{v}_i / r_{ii}$

for $j = i+1$ to n

$r_{ij} = \mathbf{q}_i^* \mathbf{v}_j$

$\mathbf{v}_j = \mathbf{v}_j - r_{ij}\mathbf{q}_i$

Modified Gram-Schmidt Orthogonalization

Classical Gram-Schmidt method:

for $j = 1$ **to** n
  $\mathbf{v}_j = \mathbf{a}_j$;
  for $i = 1$ **to** $j-1$
    $r_{ij} = \mathbf{q}_i^* \mathbf{a}_j$
    $\mathbf{v}_j = \mathbf{v}_j - r_{ij}\mathbf{q}_i$
  $r_{jj} = \|\mathbf{v}_j\|_2$
  $\mathbf{q}_j = \frac{\mathbf{v}_j}{r_{jj}}$

Modified Gram-Schmidt method:

for $j = 1$ to n
  $\mathbf{v}_j = \mathbf{a}_j$
for $i = 1$ to n
  $r_{ii} = \|\mathbf{v}_i\|_2$
  $\mathbf{q}_i = \mathbf{v}_i / r_{ii}$
  for $j = i+1$ to n
    $r_{ij} = \mathbf{q}_i^* \mathbf{v}_j$
    $\mathbf{v}_j = \mathbf{v}_j - r_{ij}\mathbf{q}_i$

- Key difference between CGS and MGS is how r_{ij} is computed
- CGS above is column-oriented (in the sense that R is computed column by column) and MGS above is row-oriented, but this is NOT the main difference between CGS and MGS. There are also column-oriented MGS and row-oriented CGS.
- MGS is numerically more stable than CGS (less sensitive to round-off errors)

Example: CGS vs. MGS

- Consider matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 \\ \varepsilon & 0 & 0 \\ 0 & \varepsilon & 0 \\ 0 & 0 & \varepsilon \end{bmatrix}$$

  where ε is small such that $1+\varepsilon^2 = 1$ with round-off error

- For both CGS and MGS

$$\begin{aligned} \mathbf{v}_1 \leftarrow (1,\varepsilon,0,0)^T, r_{11} = \sqrt{1+\varepsilon^2} \approx 1, \mathbf{q}_1 = \mathbf{v}_1/r_{11} = (1,\varepsilon,0,0)^T, \\ \mathbf{v}_2 \leftarrow (1,0,\varepsilon,0)^T, r_{12} = \mathbf{q}_1^T \mathbf{a}_2 (or = \mathbf{q}_1^T \mathbf{v}_2) = 1 \\ \mathbf{v}_2 \leftarrow \mathbf{v}_2 - r_{12}\mathbf{q}_1 = (0,-\varepsilon,\varepsilon,0)^T \\ r_{22} = \sqrt{2}\varepsilon, \mathbf{q}_2 = (0,-1,1,0)/\sqrt{2}, \\ \mathbf{v}_3 \leftarrow (1,0,0,\varepsilon)^T, r_{13} = \mathbf{q}_1^T \mathbf{a}_3 (or = \mathbf{q}_1^T \mathbf{v}_3) = 1 \\ \mathbf{v}_3 \leftarrow \mathbf{v}_3 - r_{13}\mathbf{q}_1 = (0,-\varepsilon,0,\varepsilon)^T \end{aligned}$$

Example: CGS vs. MGS Cont'd

- For CGS:

$$r_{23} = \mathbf{q}_2^T \mathbf{a}_3 = 0, \mathbf{v}_3 \leftarrow \mathbf{v}_3 - r_{23}\mathbf{q}_2 = (0, -\varepsilon, 0, \varepsilon)^T$$
$$r_{33} = \sqrt{2}\varepsilon, \mathbf{q}_3 = \mathbf{v}_3 / r_{33} = (0, -1, 0, 1)^T / \sqrt{2}$$

  - Note that $\mathbf{q}_2^T \mathbf{q}_3 = (0, -1, 1, 0))0, -1, 0, 1)^T / 2 = 1/2$

- For MGS:

$$r_{23} = \mathbf{q}_2^T \mathbf{a}_3 = \varepsilon/\sqrt{2}, \mathbf{v}_3 \leftarrow \mathbf{v}_3 - r_{23}\mathbf{q}_2 = (0, -\varepsilon/2, -\varepsilon/2, \varepsilon)^T$$
$$r_{33} = \sqrt{6}\varepsilon/2, \mathbf{q}_3 = \mathbf{v}_3 / r_{33} = (0, -1, -1, 2)^T / \sqrt{6}$$

  - Note that $\mathbf{q}_2^T \mathbf{q}_3 = (0, -1, 1, 0)(0, -1, -1, 2)^T / \sqrt{12} = 0$

Operation Count

- It is important to assess the efficiency of algorithms. But how?
  - We could implement different algorithms and do head-to-head comparison, but implementation details might affect true performance
  - We could estimate cost of all operations, but it is very tedious
  - Relatively simple and effective approach is to estimate amount of floating-point operations, or 'flops', and focus on asymptotic analysis as sizes of matrices approach infinity
- Count each operation $+, -, *, /$, and $\sqrt{}$ as one flop, and make no distinction of real and complex numbers

Theorem

CGS and MGS require $\sim 2mn^2$ flops to compute a QR factorization of an $m \times n$ matrix.

Outline

Householder Reflectors

Gram-Schmidt as Triangular Orthogonalization

- Every step of Gram-Schmidt can be viewed as multiplication with triangular matrix. For example, at first step:

$$[\mathbf{v}_1|\mathbf{v}_2|\dots\mathbf{v}_n]\underbrace{\begin{bmatrix}\frac{1}{r_{11}} & \frac{-r_{12}}{r_{11}} & \frac{-r_{13}}{r_{11}} & \dots \\ & 1 & & \\ & & 1 & \\ & & & \ddots\end{bmatrix}}_{\mathbf{R}_1} = [q_1|\mathbf{v}_2^{(2)}|\dots|\mathbf{v}_n^{(2)}],$$

- Gram-Schmidt therefore multiplies triangular matrices to orthogonalize column vectors, and in turns can be viewed as triangular orthogonalization

$$\underbrace{\mathbf{A}\mathbf{R}_1\mathbf{R}_2\dots\mathbf{R}_n}_{\hat{\mathbf{R}}^{-1}} = \hat{\mathbf{Q}}$$

  where $\mathbf{R}_i$ is a triangular matrix.
- A 'dual' approach would be orthogonal triangularization, i.e., multiply $\mathbf{A}$ by unitary matrices to make it triangular matrix

Householder Triangularization

- Method introduced by Alston Scott Householder in 1958
- It multiplies unitary matrices to make column triangular, e.g.

$$\underbrace{\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}}_{\mathbf{A}} \xrightarrow{\mathbf{Q}_1} \underbrace{\begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix}}_{\mathbf{Q}_1\mathbf{A}} \xrightarrow{\mathbf{Q}_2} \underbrace{\begin{bmatrix} \times & \times & \times \\ & \times & \times \\ & 0 & \times \\ & 0 & \times \\ & 0 & \times \end{bmatrix}}_{\mathbf{Q}_2\mathbf{Q}_1\mathbf{A}} \xrightarrow{\mathbf{Q}_3} \underbrace{\begin{bmatrix} \times & \times & \times \\ & \times & \times \\ & & \times \\ & & 0 \\ & & 0 \end{bmatrix}}_{\mathbf{Q}_3\mathbf{Q}_2\mathbf{Q}_1\mathbf{A}}$$

- After n steps, we get a product of unitary matrices

$$\underbrace{\mathbf{Q}_n \dots \mathbf{Q}_2 \mathbf{Q}_1}_{\mathbf{Q}^*} \mathbf{A} = \mathbf{R}$$

  and in turn we get full QR factorization $\mathbf{A} = \mathbf{QR}$
- $\mathbf{Q}_k$ introduces zeros below diagonal of kth column while preserving zeros below diagonal in preceding columns
- The key question is how to find $\mathbf{Q}_k$

Householder Reflectors

- First, consider $\mathbf{Q}_1 : \mathbf{Q}_1\mathbf{a}_1 = \|\mathbf{a}_1\|\mathbf{e}_1$, where $\mathbf{e}_1 = (1,0,\ldots,0)^T$. Why the length is $\|\mathbf{a}_1\|$?
- $\mathbf{Q}_1$ reflects a_1 across hyperplane H orthogonal to $\mathbf{v} = \|a_1\|e_1 - a_1$, and therefore

$$\mathbf{Q}_1 = \mathbf{I} - 2\frac{\mathbf{v}\mathbf{v}^*}{\mathbf{v}^*\mathbf{v}}$$

- More generally,

$$\mathbf{Q}_k = \begin{bmatrix} \mathbf{I} & 0 \\ 0 & \mathbf{F} \end{bmatrix}$$

  where $\mathbf{I}$ is $(k-1)\times(k-1)$ and $\mathbf{F}$ is $(m-k+1)\times(m-k+1)$ such that $\mathbf{F}\mathbf{x} = \|x\|_2 e_1$, where $\mathbf{x}$ is $(a_{k,k}, a_{k,k+1}, \ldots, a_{k,m})^T$
- What is $\mathbf{F}$? It has similar form as $\mathbf{Q}_1$ with $\mathbf{v} = \|x\|e_1 - \mathbf{x}$.

Choice of Reflectors

- We could choose $\mathbf{F}$ such that $\mathbf{Fx} = -\|x\| e_1$ instead of $\mathbf{Fx} = \|x\| e_1$, or more generally, $\mathbf{Fx} = z\|\mathbf{x}\| e_1$ with $|z| = 1$ for $z \in \mathbb{C}$
- This leads to an infinite number of possible QR factorizations of $\mathbf{A}$
- If we require $z \in \mathbb{R}$, we still have two choices
- Numerically, it is undesirable for $\mathbf{v}^*\mathbf{v}$ to be close to zero for $\mathbf{v} = z\|\mathbf{x}\| e_1 - \mathbf{x}$, and $\|v\|$ is larger if $z = -sign(x_1)$
- Therefore, $\mathbf{v} = -sign(x_1)\|\mathbf{x}\| e_1 - \mathbf{x}$. Since $\mathbf{I} - 2\frac{\mathbf{vv}^*}{\mathbf{v}^*\mathbf{v}}$ is independent of sign, clear out the factor -1 and obtain $\mathbf{v} = sign(x_1)\|\mathbf{x}\| e_1 + \mathbf{x}$
- For completeness, if $x_1 = 0$, set z to 1 (instead of 0)

Householder Algorithm

Householder QR Factorization

for $k = 1$ to n
$\mathbf{x} = \mathbf{A}_{k:m,k}$
$\mathbf{v}_k = sign(x_1)\,\|x\|\,e_1 + \mathbf{x}$
$\mathbf{v}_k = \mathbf{v}_k / \|\mathbf{v}_k\|$
$\mathbf{A}_{k:m,k:n} = \mathbf{A}_{k:m,k:n} - 2\mathbf{v}_k(\mathbf{v}_k^* \mathbf{A}_{k:m,k:n})$

- Note that $sign(x) = 1$ if $\mathbf{x} \geq 0$ and $= -1$ if $\mathbf{x} < 0$
- Leave $\mathbf{R}$ in place of $\mathbf{A}$
- Matrix $\mathbf{Q}$ is not formed explicitly but reflection vector $\mathbf{v}_k$ is stored

Householder Algorithm

Householder QR Factorization

for $k = 1$ to n
$\mathbf{x} = \mathbf{A}_{k:m,k}$
$\mathbf{v}_k = sign(x_1)\,\|x\|\,e_1 + \mathbf{x}$
$\mathbf{v}_k = \mathbf{v}_k / \|\mathbf{v}_k\|$
$\mathbf{A}_{k:m,k:n} = \mathbf{A}_{k:m,k:n} - 2\mathbf{v}_k(\mathbf{v}_k^* \mathbf{A}_{k:m,k:n})$

- Note that $sign(x) = 1$ if $\mathbf{x} \geq 0$ and $= -1$ if $\mathbf{x} < 0$
- Leave $\mathbf{R}$ in place of $\mathbf{A}$
- Matrix $\mathbf{Q}$ is not formed explicitly but reflection vector $\mathbf{v}_k$ is stored
- Question: Can $\mathbf{A}$ be reused to store both $\mathbf{R}$ and $\mathbf{v}_k$ completely?

Householder Algorithm

Householder QR Factorization

for $k = 1$ to n
$\mathbf{x} = \mathbf{A}_{k:m,k}$
$\mathbf{v}_k = sign(x_1)\,\|x\|\,e_1 + \mathbf{x}$
$\mathbf{v}_k = \mathbf{v}_k / \|\mathbf{v}_k\|$
$\mathbf{A}_{k:m,k:n} = \mathbf{A}_{k:m,k:n} - 2\mathbf{v}_k(\mathbf{v}_k^* \mathbf{A}_{k:m,k:n})$

- Note that $sign(x) = 1$ if $\mathbf{x} \geq 0$ and $= -1$ if $\mathbf{x} < 0$
- Leave $\mathbf{R}$ in place of $\mathbf{A}$
- Matrix $\mathbf{Q}$ is not formed explicitly but reflection vector $\mathbf{v}_k$ is stored
- Question: Can $\mathbf{A}$ be reused to store both $\mathbf{R}$ and $\mathbf{v}_k$ completely?
- Answer: We can use lower diagonal portion of $\mathbf{A}$ to store all but one entry in each $\mathbf{v}_k$. So an additional array of size n is needed.

Householder Algorithm

Householder QR Factorization

for $k = 1$ to n
$\mathbf{x} = \mathbf{A}_{k:m,k}$
$\mathbf{v}_k = sign(x_1)\,\|x\|\,e_1 + \mathbf{x}$
$\mathbf{v}_k = \mathbf{v}_k / \|\mathbf{v}_k\|$
$\mathbf{A}_{k:m,k:n} = \mathbf{A}_{k:m,k:n} - 2\mathbf{v}_k(\mathbf{v}_k^* \mathbf{A}_{k:m,k:n})$

- Note that $sign(x) = 1$ if $\mathbf{x} \geq 0$ and $= -1$ if $\mathbf{x} < 0$
- Leave $\mathbf{R}$ in place of $\mathbf{A}$
- Matrix $\mathbf{Q}$ is not formed explicitly but reflection vector $\mathbf{v}_k$ is stored
- Question: Can $\mathbf{A}$ be reused to store both $\mathbf{R}$ and $\mathbf{v}_k$ completely?
- Answer: We can use lower diagonal portion of $\mathbf{A}$ to store all but one entry in each $\mathbf{v}_k$. So an additional array of size n is needed.
- Question: What happens if $\mathbf{v}_k$ is 0 in line 3 of the loop?

Applying or Forming $\mathbf{Q}$

- Compute $\mathbf{Q}^*\mathbf{b} = \mathbf{Q}_n \dots \mathbf{Q}_1\mathbf{b}$

  Implicit calculation of $\mathbf{Q}^*\mathbf{b}$
  for $k = 1$ to n
  $b_{k:m} = b_{k:m} - 2\mathbf{v}_k(\mathbf{v}_k^*\mathbf{b}_{k:m})$

- Compute $\mathbf{Q}\mathbf{x} = \mathbf{Q}_1\mathbf{Q}_2 \dots \mathbf{Q}_n\mathbf{x}$

  Implicit calculation of $\mathbf{Q}\mathbf{x}$
  for $k = n$ down to 1
  $\mathbf{x}_{k:m} = \mathbf{x}_{k:m} - 2\mathbf{v}_k(\mathbf{v}_k^*\mathbf{x}_{k:m})$

- Question: How to form $\mathbf{Q}$ and $\hat{\mathbf{Q}}$, respectively?
- Answer: Apply $\mathbf{x} = \mathbf{I}_{m\times m}$ or first n columns of $\mathbf{I}$, respectively

Operation Count

- Most work done at step $\mathbf{A}_{k:m,k:n} = \mathbf{A}_{k:m,k:n} - 2\mathbf{v}_k(\mathbf{v}_k^*\mathbf{A}_{k:m,k:n})$
- Flops per iteration:
  - $\sim 2(m-k)(n-k)$ for dot products $\mathbf{v}_k^*\mathbf{A}_{k:m,k:n}$
  - $\sim (m-k)(n-k)$ for outer product $2v_k(\ldots)$
  - $\sim (m-k)(n-k)$ for subtraction
  - $\sim 4(m-k)(n-k)$ total
- Including outer loop, total flops is

$$\begin{aligned}\sum_{k=1}^{n} 4(m-k)(n-k) &= 4\sum_{k=1}^{n}(mn-km-kn+k^2)\\ &\sim 4mn^2 - 4(m+n)n^2/2 + 4n^3/3\\ &= 2mn^2 - \frac{2}{3}n^3\end{aligned}$$

If $m \approx n$, it is more efficient than Gram-Schmidt method, but if $m \gg n$, similar to Gram-Schmidt

Givens Rotations

- Instead of using reflection, we can rotate $\mathbf{x}$ to obtain $\|x\| e_1$
- A Given rotation $\mathbf{R} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$ rotates $\mathbf{x} \in \mathbb{R}^2$ counterclockwise by θ
- Choose θ to be angle between $(x_i, x_j)^T$ and $(1,0)^T$, and we have

$$\begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x_i \\ x_j \end{bmatrix} = \begin{bmatrix} \sqrt{x_i^2 + x_j^2} \\ 0 \end{bmatrix}$$

where

$$cos\theta = \frac{x_i}{\sqrt{x_i^2 + x_j^2}}, \quad sin\theta = \frac{-x_j}{\sqrt{x_i^2 + x_j^2}}$$

Givens QR

- Introduce zeros in column bottom-up, one zero at a time

$$\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \xrightarrow{(4,5)} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \end{bmatrix} \xrightarrow{(3,4)} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \\ & \times & \times \end{bmatrix} \dots$$

- To zero a_{ij} , left-multiply matrix $\mathbf{F}$ with $\mathbf{F}_{i:i+1,i:i+1}$ being rotation matrix and $\mathbf{F}_{kk} = 1$ for $k \neq i, i+1$
- Flop count of Givens **QR** is $3mn^2 - n^3$, which is about 50% more expensive than Householder QR

Linear Least Squares Problems

Linear Least Squares Problems

- Overdetermined system of equations $\mathbf{Ax} \approx \mathbf{b}$, where $\mathbf{A}$ has more rows than columns and has full rank, in general has no solutions
- Example application: Polynomial least squares fitting
- In general, minimize the residual $\mathbf{r} = \mathbf{b} - \mathbf{Ax}$
- In terms of 2-norm, we obtain linear least squares problem: Given $\mathbf{A} \in \mathbb{C}^{m \times n}$, $m \geq n$, and $\mathbf{b} \in \mathbb{C}^m$, find $\mathbf{x} \in \mathbb{C}^n$ such that $\|\mathbf{b} - \mathbf{Ax}\|_2$ is minimized
- If $\mathbf{A}$ has full rank, the minimizer $\mathbf{x}$ is the solution to the normal equation

$$\mathbf{A}^*\mathbf{Ax} = \mathbf{A}^*\mathbf{b}$$

or in terms of the pseudoinverse $\mathbf{A}^+$,

$$\mathbf{x} = \mathbf{A}^+\mathbf{b}, \text{ where } \mathbf{A}^+ = (\mathbf{A}^*\mathbf{A})^{-1}\mathbf{A}^* \in \mathbb{C}^{n \times m}$$

Geometric Interpretation

- $\mathbf{Ax}$ is in range($\mathbf{A}$), and the point in range($\mathbf{A}$) closest to $\mathbf{b}$ is its orthogonal projection onto range($\mathbf{A}$)
- Residual $\mathbf{r}$ is then orthogonal to range($\mathbf{A}$), and hence $\mathbf{A}^*\mathbf{r} = \mathbf{A}^*(\mathbf{b} - \mathbf{Ax}) = 0$
- $\mathbf{Ax}$ is orthogonal projection of $\mathbf{b}$, where $\mathbf{x} = \mathbf{A}^+\mathbf{b}$, so $\mathbf{P} = \mathbf{AA}^+ = \mathbf{A}(\mathbf{A}^*\mathbf{A})^{-1}\mathbf{A}^*$ is orthogonal projection (recall lecture 5)

Solution of Least Squares Problems

- One approach is to solve normal equation $\mathbf{A}^*\mathbf{A}\mathbf{x} = \mathbf{A}^*\mathbf{b}$ directly using Cholesky factorization
  - Is unstable, but is very efficient if $m \gg n$ $(mn^2 + \frac{1}{3}n^3)$
- More robust approach is to use QR factorization $\mathbf{A} = \hat{\mathbf{Q}}\hat{\mathbf{R}}$
  - $\mathbf{b}$ can be projected onto range($\mathbf{A}$) by $\mathbf{P} = \hat{\mathbf{Q}}\hat{\mathbf{Q}}^*$, and therefore $\hat{\mathbf{Q}}\hat{\mathbf{R}}\mathbf{x} = \hat{\mathbf{Q}}\hat{\mathbf{Q}}^*\mathbf{b}$
  - Left-multiply by $\hat{\mathbf{Q}}^*$ and we get $\hat{\mathbf{R}}\mathbf{x} = \hat{\mathbf{Q}}^*\mathbf{b}$(note $\mathbf{A}^+ = \hat{\mathbf{R}}^{-1}\hat{\mathbf{Q}}^*$)

> Least squares via QR Factorization:
> Compute reduced QR factorization $\mathbf{A} = \hat{\mathbf{Q}}\hat{\mathbf{R}}$
> Compute vector $\mathbf{c} = \hat{\mathbf{Q}}^*\mathbf{b}$
> Solve upper-triangular system $\hat{\mathbf{R}}\mathbf{x} = \mathbf{c}$ for $\mathbf{x}$

Solution of Least Squares Problems contd.

Least squares via QR Factorization:
Compute reduced QR factorization $\mathbf{A} = \hat{\mathbf{Q}}\hat{\mathbf{R}}$
Compute vector $\mathbf{c} = \hat{\mathbf{Q}}^*\mathbf{b}$
Solve upper-triangular system $\hat{\mathbf{R}}\mathbf{x} = \mathbf{c}$ for $\mathbf{x}$

- Computation is dominated by QR factorization ($2mn^2 - \frac{2}{3}n^3$)
- Question: If Householder QR is used, how to compute $\hat{\mathbf{Q}}\mathbf{b}$?
- Answer: Compute $\mathbf{Q}^*\mathbf{b}$ (where $\mathbf{Q}$ is from full QR factorization) and then take first n entries of resulting $\mathbf{Q}^*\mathbf{b}$

Solution by SVD

- Using $\mathbf{A} = \hat{\mathbf{U}}\hat{\Sigma}\mathbf{V}^*$, $\mathbf{b}$ can be projected onto range($\mathbf{A}$) by $\mathbf{P} = \hat{\mathbf{U}}\hat{\mathbf{U}}^*$, and therefore $\hat{\mathbf{U}}\hat{\Sigma}\mathbf{V}^*\mathbf{x} = \hat{\mathbf{U}}\hat{\mathbf{U}}^*\mathbf{b}$
- Left-multiply by $\hat{\mathbf{U}}$ and we get $\hat{\Sigma}\hat{\mathbf{V}}^*\mathbf{x} = \hat{\mathbf{U}}^*\mathbf{b}$

> Least squares via SVD:
> Compute reduced SVD factorization $\mathbf{A} = \hat{\mathbf{U}}\hat{\Sigma}\mathbf{V}^*$
> Compute vector $\mathbf{c} = \hat{\mathbf{U}}^*\mathbf{b}$
> Solve diagonal system $\hat{\Sigma}\mathbf{w} = \mathbf{c}$ for $\mathbf{w}$
> Set $\mathbf{x} = \mathbf{V}\mathbf{w}$

- Work is dominated by SVD, which is $\sim 2mn^2 + 11n^3$ flops, very expensive if $m \approx n$
- Best numerical stability
- Question: If $\mathbf{A}$ is rank deficient, how to solve $\mathbf{A}\mathbf{x} \approx \mathbf{b}$?

Solution by SVD

- Using $\mathbf{A} = \hat{\mathbf{U}}\hat{\Sigma}\mathbf{V}^*$, $\mathbf{b}$ can be projected onto range($\mathbf{A}$) by $\mathbf{P} = \hat{\mathbf{U}}\hat{\mathbf{U}}^*$, and therefore $\hat{\mathbf{U}}\hat{\Sigma}\mathbf{V}^*\mathbf{x} = \hat{\mathbf{U}}\hat{\mathbf{U}}^*\mathbf{b}$
- Left-multiply by $\hat{\mathbf{U}}$ and we get $\hat{\Sigma}\hat{\mathbf{V}}^*\mathbf{x} = \hat{\mathbf{U}}^*\mathbf{b}$

> Least squares via SVD:
> Compute reduced SVD factorization $\mathbf{A} = \hat{\mathbf{U}}\hat{\Sigma}\mathbf{V}^*$
> Compute vector $\mathbf{c} = \hat{\mathbf{U}}^*\mathbf{b}$
> Solve diagonal system $\hat{\Sigma}\mathbf{w} = \mathbf{c}$ for $\mathbf{w}$
> Set $\mathbf{x} = \mathbf{V}\mathbf{w}$

- Work is dominated by SVD, which is $\sim 2mn^2 + 11n^3$ flops, very expensive if $m \approx n$
- Best numerical stability
- Question: If $\mathbf{A}$ is rank deficient, how to solve $\mathbf{A}\mathbf{x} \approx \mathbf{b}$?
- Answer: $\mathbf{x}$ is no longer unique. Constrain $\mathbf{x}$ to be orthogonal to null space of $\mathbf{A}$.

Outline

Conditioning and Condition Numbers

Overview of Error Analysis

- Error analysis is important subject of numerical analysis
- Given a problem f and an algorithm $\tilde{f}$ with an input x, the absolute error is $\left\|\tilde{f}(x) - f(x)\right\|$ and relative error is $\left\|\tilde{f}(x) - f(x)\right\| / \|f(x)\|$
- What are possible sources of errors?

Overview of Error Analysis

- Error analysis is important subject of numerical analysis
- Given a problem f and an algorithm $\tilde{f}$ with an input x, the absolute error is $\left\|\tilde{f}(x) - f(x)\right\|$ and relative error is $\left\|\tilde{f}(x) - f(x)\right\| / \|f(x)\|$
- What are possible sources of errors?
  - Round-off error (input, computation), truncation (approximation) error
- We would like the solution to be accurate, i.e., with small errors
- The error depends on property (conditioning) of the problem, property (stability) of the algorithm

Absolute Condition Number

- Condition number is a measure of sensitivity of a problem
- Absolute condition number of a problem $\mathbf{f}$ at $\mathbf{x}$ is

$$\hat{\kappa} = \lim_{\varepsilon \to 0} sup_{\|\delta \mathbf{x}\| \leq \varepsilon} \frac{\|\delta \mathbf{f}\|}{\|\delta \mathbf{X}\|}$$

  where $\delta \mathbf{f} = \mathbf{f}(\mathbf{x} + \delta \mathbf{x}) - \mathbf{f}(\mathbf{x})$
- Less formally, $\hat{\kappa} = sup_{\delta \mathbf{x}} \frac{\|\delta \mathbf{f}\|}{\|\delta \mathbf{X}\|}$ for infinitesimally small $\delta \mathbf{x}$
- If $\mathbf{f}$ is differentiable, then

$$\hat{\kappa} = \|\mathbf{J}(x)\|$$

  where $\mathbf{J}$ is the Jacobian of $\mathbf{f}$ at $\mathbf{x}$, with $J_{ij} = \frac{\delta f_i}{\delta x_j}$, and the matrix norm is induced by vector norms on $\delta \mathbf{f}$ and $\delta \mathbf{x}$
- Question: What is absolute condition number of $\mathbf{f}(\mathbf{x}) = \alpha \mathbf{x}$?
- Question: Is absolute condition number scale invariant?

Relative Condition Number

- Relative condition number of a problem $\mathbf{f}$ at $\mathbf{x}$ is

$$\hat{\kappa} = \lim_{\varepsilon \to 0} sup_{\|\delta \mathbf{x}\| \leq \varepsilon} \frac{\|\delta \mathbf{f}\| / \|\mathbf{f}(\mathbf{x})\|}{\|\delta \mathbf{X}\| / \|\mathbf{x}\|}$$

- Less formally, $\hat{\kappa} = sup_{\delta \mathbf{X}} \frac{\|\delta \mathbf{f}\|/\|\mathbf{f}(\mathbf{x})\|}{\|\delta \mathbf{X}\|/\|\mathbf{x}\|}$ for infinitesimally small $\delta \mathbf{x}$
- Note: we can use different types of norms to get different condition numbers
- If $\mathbf{f}$ is differentiable, then

$$\hat{\kappa} = \frac{\|\mathbf{J}(x)\|}{\|\mathbf{f}(\mathbf{x})\| / \|\mathbf{x}\|}$$

- Question: What is relative condition number of $\mathbf{f}(\mathbf{x}) = \alpha \mathbf{x}$?
- Question: Is absolute condition number scale invariant?
- In numerical analysis, we in general use relative condition number
- A problem is well-conditioned if κ is small and is ill-conditioned if κ is large

Condition Numbers

- Absolute condition number of a problem $\mathbf{f}$ at $\mathbf{x}$ is

$$\hat{\kappa} = \lim_{\varepsilon \to 0} sup_{\|\delta \mathbf{x}\| \leq \varepsilon} \frac{\|\delta \mathbf{f}\|}{\|\delta \mathbf{X}\|}$$

  where $\delta \mathbf{f} = \mathbf{f}(\mathbf{x} + \delta \mathbf{x}) - \mathbf{f}(\mathbf{x})$

- Less formally, $\hat{\kappa} = sup_{\delta \mathbf{X}} \frac{\|\delta \mathbf{f}\|}{\|\delta \mathbf{X}\|}$ for infinitesimally small $\delta \mathbf{x}$

- Relative condition number of a problem $\mathbf{f}$ at $\mathbf{x}$ is

$$\hat{\kappa} = \lim_{\varepsilon \to 0} sup_{\|\delta \mathbf{x}\| \leq \varepsilon} \frac{\|\delta \mathbf{f}\| / \|\mathbf{f}(\mathbf{x})\|}{\|\delta \mathbf{X}\| / \|\mathbf{x}\|}$$

- Less formally, $\hat{\kappa} = sup_{\delta \mathbf{X}} \frac{\|\delta \mathbf{f}\|/\|\mathbf{f}(\mathbf{x})\|}{\|\delta \mathbf{X}\|/\|\mathbf{x}\|}$ for infinitesimally small $\delta \mathbf{x}$

Examples

- Example: Function $f(x) = \sqrt{x}$
  - Absolute condition number of $\mathbf{f}$ at $\mathbf{x}$ is $\hat{\kappa} = \|\mathbf{J}\| = 1/(2\mathbf{x})$
  - Relative condition number $\hat{\kappa} = \frac{\|\mathbf{J}(x)\|}{\|\mathbf{f}(\mathbf{x})\|/\|\mathbf{x}\|} = \frac{1/(2\sqrt{\mathbf{x}}}{\sqrt{\mathbf{x}}/\mathbf{x}} = 1/2$
- Example: Function $f(x) = x_1 - x_2$, where $x = (x_1, x_2)^T$
  - Absolute condition number of f at x in ∞-norm is $\hat{\kappa} = \|\mathbf{J}\|_\infty = \|(1, -1)\|_\infty = 2$
  - Relative condition number$\hat{\kappa} = \frac{\|\mathbf{J}(x)\|}{\|\mathbf{f}(\mathbf{x})\|/\|\mathbf{x}\|} = \frac{2}{|x_1 - x_2|/max\{|x_1|,|x_2|\}}$
  - κ is arbitrarily large (f is ill-conditioned) if $x_1 \approx x_2$ (hazard of cancellation error)
- Note: From now on, we will talk about only relative condition number

Condition Number of Matrix-Vector Product

- Consider $\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x}$, with $\mathbf{A} \in \mathbb{C}^{m \times n}$

$$\hat{\kappa} = \frac{\|\mathbf{J}(x)\|}{\|\mathbf{f}(\mathbf{x})\| / \|\mathbf{x}\|} = \frac{\|\mathbf{A}\| \, \|\mathbf{x}\|}{\|\mathbf{A}\mathbf{x}\|}$$

- If $\mathbf{A}$ is square and nonsingular, since $\|\mathbf{x}\| / \|\mathbf{A}\mathbf{x}\| \leq \left\|\mathbf{A}^{-1}\right\|$

$$\kappa \leq \|\mathbf{A}\| \left\|\mathbf{A}^{-1}\right\|$$

  - Question: For what $\mathbf{x}$ is equality achieved if 2-norm is used?

Condition Number of Matrix-Vector Product

- Consider $\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x}$, with $\mathbf{A} \in \mathbb{C}^{m \times n}$

$$\hat{\kappa} = \frac{\|\mathbf{J}(x)\|}{\|\mathbf{f}(\mathbf{x})\| / \|\mathbf{x}\|} = \frac{\|\mathbf{A}\| \|\mathbf{x}\|}{\|\mathbf{A}\mathbf{x}\|}$$

- If $\mathbf{A}$ is square and nonsingular, since $\|\mathbf{x}\| / \|\mathbf{A}\mathbf{x}\| \leq \left\|\mathbf{A}^{-1}\right\|$

$$\kappa \leq \|\mathbf{A}\| \left\|\mathbf{A}^{-1}\right\|$$

  - Question: For what $\mathbf{x}$ is equality achieved if 2-norm is used?
  - Answer: $\mathbf{x}$ is equal to right singular vector corresponding to smallest singular value of $\mathbf{A}$
  - Question: What is condition number of $\mathbf{A}\mathbf{x}$ if $\mathbf{A}$ is singular?
  - Answer: $\leq \infty$ (is ∞ if $\mathbf{x} \in \mathit{null}(\mathbf{A})$).
- What is the condition number for $f(\mathbf{b}) = \mathbf{A}^{-1}\mathbf{b}$?

Condition Number of Matrix-Vector Product

- Consider $\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x}$, with $\mathbf{A} \in \mathbb{C}^{m \times n}$

$$\hat{\kappa} = \frac{\|\mathbf{J}(x)\|}{\|\mathbf{f}(\mathbf{x})\| / \|\mathbf{x}\|} = \frac{\|\mathbf{A}\| \|\mathbf{x}\|}{\|\mathbf{A}\mathbf{x}\|}$$

- If $\mathbf{A}$ is square and nonsingular, since $\|\mathbf{x}\| / \|\mathbf{A}\mathbf{x}\| \leq \left\|\mathbf{A}^{-1}\right\|$

$$\kappa \leq \|\mathbf{A}\| \left\|\mathbf{A}^{-1}\right\|$$

  - Question: For what $\mathbf{x}$ is equality achieved if 2-norm is used?
  - Answer: $\mathbf{x}$ is equal to right singular vector corresponding to smallest singular value of $\mathbf{A}$
  - Question: What is condition number of $\mathbf{A}\mathbf{x}$ if $\mathbf{A}$ is singular?
  - Answer: $\leq \infty$ (is ∞ if $\mathbf{x} \in \mathit{null}(\mathbf{A})$).
- What is the condition number for $f(\mathbf{b}) = \mathbf{A}^{-1}\mathbf{b}$?
  - Answer: $\kappa \leq \|\mathbf{A}\| \left\|\mathbf{A}^{-1}\right\|$

Condition Number of a Matrix

- We define condition number of matrix $\mathbf{A}$ as

$$\kappa \leq \|\mathbf{A}\| \left\|\mathbf{A}^{-1}\right\|$$

- It is the upper bound of the condition number of $\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x}$ for any $\mathbf{x}$
- Another way to interpret at $\kappa(\mathbf{A})$ is

$$\kappa(\mathbf{A}) = sup_{\delta\mathbf{x},\mathbf{x}} \frac{\|\delta\mathbf{f}\| / \|\delta\mathbf{x}\|}{\|\mathbf{f}(\mathbf{x})\| / \|\mathbf{x}\|} = \frac{sup_{\delta\mathbf{x}} \|\mathbf{A}\delta\mathbf{x}\| / \|\delta\mathbf{x}\|}{inf_{\mathbf{x}} \|\mathbf{A}\mathbf{x}\| / \|\mathbf{x}\|}$$

- For 2-norm, $\kappa(\mathbf{A}) = \frac{\sigma_1}{\sigma_n}$
- Note about the distinction between the condition number of a problem (the map $\mathbf{f}(\mathbf{x})$) and the condition number of a problem instance (the evaluation of $\mathbf{f}(\mathbf{x})$ for specific $\mathbf{x}$)
- Note: condition number of a problem is a property of a problem, and is independent of its algorithm

Accuracy and Stability

Accuracy

- Roughly speaking, accuracy means that "error" is small in an asymptotic sense, say $O(\varepsilon_{machine})$
- When we say $O(\varepsilon_{machine})$, we are thinking of a series of idealized machines for which $\varepsilon_{machine}$ can be arbitrarily small

More on Accuracy

- An algorithm $\tilde{\mathbf{f}}$ is accurate if relative error is in the order of machine precision, i.e.,

$$\left\|\tilde{\mathbf{f}}(\mathbf{x}) - \mathbf{f}(\mathbf{x})\right\| / \left\|\mathbf{f}(\mathbf{x})\right\| = O(\varepsilon_{machine}),$$

  i.e., $\leq C_1 \varepsilon_{machine}$ as $\varepsilon_{machine} \to 0$, where constant C_1 may depend on the condition number and the algorithm itself

- In most cases, we expect

$$\left\|\tilde{\mathbf{f}}(\mathbf{x}) - \mathbf{f}(\mathbf{x})\right\| / \left\|\mathbf{f}(\mathbf{x})\right\| = O(\kappa \varepsilon_{machine}),$$

  i.e., $\leq C \kappa \varepsilon_{machine}$ as $\varepsilon_{machine} \to 0$, where constant C should be independent of κ and value of $\mathbf{x}$ (although it may depend on the dimension of $\mathbf{x}$)

More on Accuracy

- How do we determine whether an algorithm is accurate or not?
  - It turns out to be an extremely subtle question
  - A forward error analysis (operation by operation) is often too difficult and impractical, and cannot capture dependence on condition number
  - An effective solution is **backward error analysis**

Stability

Stable Algorithm

- Nearly the right answer to nearly the right question
- More formally, an algorithm $\tilde{\mathbf{f}}$ for problem $\mathbf{f}$ is stable if (for all $\mathbf{x}$)

$$\left\|\tilde{\mathbf{f}}(\mathbf{x}) - f(\tilde{\mathbf{x}})\right\| / \left\|f(\tilde{\mathbf{x}})\right\| = O(\varepsilon_{machine}),$$

Stability

Stable Algorithm

- Nearly the right answer to nearly the right question
- More formally, an algorithm $\tilde{\mathbf{f}}$ for problem $\mathbf{f}$ is stable if (for all $\mathbf{x}$)

$$\left\|\tilde{\mathbf{f}}(\mathbf{x}) - f(\tilde{\mathbf{x}})\right\| / \left\|f(\tilde{\mathbf{x}})\right\| = O(\varepsilon_{machine}),$$

Backward Stable Algorithm

- Exactly the right answer to nearly the right question
- More formally, an algorithm $\tilde{\mathbf{f}}$ for problem $\mathbf{f}$ is stable if (for all $\mathbf{x}$)

$$\tilde{\mathbf{f}}(\mathbf{x}) = \mathbf{f}(\tilde{\mathbf{x}})$$

- for some $\tilde{\mathbf{x}}$ with $\|\tilde{\mathbf{x}} - \mathbf{x}\| / \|\mathbf{x}\| = O(\varepsilon_{machine})$
- Backward stability is stronger! And it does not depend on condition number of

Stability of Floating Point Arithmetic

- Backward stability of floating point operations is implied by these two floating point axioms:
  1. $\forall x \in \mathbb{R}, \exists \varepsilon, |\varepsilon| \leq \varepsilon_{machine}$ s.t. $fl(x) = x(1+\varepsilon)$
  2. For floating-point numbers $x, y, \exists \varepsilon, |\varepsilon| \leq \varepsilon_{machine}$ s.t. $x \circledast y = (x * y)(1+\varepsilon)$

Stability of Floating Point Arithmetic

- Example: Subtraction $f(x_1, x_2) = x_1 - x_2$ with floating-point operation

$$\tilde{f}(x_1, x_2) = fl(x_1) \ominus fl(x_2)$$

  - Axiom 1 implies $fl(x_1) = x_1(1+\varepsilon_1), fl(x_2) = x_2(1+\varepsilon_2)$, for some $|\varepsilon_1|, |\varepsilon_2| \leq \varepsilon_{machine}$
  - Axiom 2 implies $fl(x_1) \ominus fl(x_2) = (fl(x_1) - fl(x_2))(1+\varepsilon_3)$ for some $|\varepsilon_3| \leq \varepsilon_{machine}$
  - Therefore,

$$\begin{aligned} fl(x_1) \ominus fl(x_2) &= (x_1(1+\varepsilon_1) - x_2(1+\varepsilon_2))(1+\varepsilon_3) \\ &= x_1(1+\varepsilon_1)(1+\varepsilon_3) - x_2(1+\varepsilon_2)(1+\varepsilon_3) \\ &= x_1(1+\varepsilon_4) - x_2(1+\varepsilon_5) \end{aligned}$$

  where $|\varepsilon_4|, |\varepsilon_5| \leq 2\varepsilon_{machine} + O(\varepsilon_{machine}^2)$

Stability of Floating Point Arithmetic Cont'd

- Example: Inner product $f(\mathbf{x},\mathbf{y}) = \mathbf{x}^*\mathbf{y}$ using floating-point operations $\otimes$ and $\oplus$ is backward stable
- Example: Outer product $f(\mathbf{x},\mathbf{y}) = \mathbf{x}\mathbf{y}^*$ using $\otimes$ and $\oplus$ is not backward stable
- Example: $f(x) = x+1$ computed as $\tilde{\mathbf{f}}(x) = fl(x) \oplus 1$ is not backward stable
- Example: $f(x,y) = x+y$ computed as $\tilde{f}(x,y) = fl(x) \oplus fl(y)$ is backward stable

Accuracy of Backward Stable Algorithm

Theorem

If a backward stable algorithm $\tilde{\mathbf{f}}$ is used to solve a problem f with condition number κ using floating-point numbers satisfying the two axioms, then

$$\left\|\tilde{\mathbf{f}}(\mathbf{x}) - f(\tilde{\mathbf{x}})\right\| / \left\|\mathbf{f}(\mathbf{x})\right\| = O(\kappa(\mathbf{x})\varepsilon_{machine})$$

Proof: Not Included - Compensation against online classes!

Backward Stability of Householder QR

- For a QR factorization $\mathbf{A} = \mathbf{QR}$ computed by Householder triangularization, the factors $\tilde{\mathbf{Q}}$ and $\tilde{\mathbf{R}}$ satisfy

$$\tilde{\mathbf{Q}}\tilde{\mathbf{R}} = \mathbf{A} + \delta\mathbf{A}, \qquad \|\delta\mathbf{A}\| / \|\mathbf{A}\| = O(\varepsilon_{machine}),$$

  i.e. exact QR factorization of a slightly perturbed $\mathbf{A}$ (we will not prove it in class)
- $\tilde{\mathbf{R}}$ is $\mathbf{R}$ computed by algorithm using floating points
- However, $\tilde{\mathbf{Q}}$ is product of exactly unitary reflectors

$$\tilde{\mathbf{Q}} = \tilde{\mathbf{Q}}_1\tilde{\mathbf{Q}}_2\cdots\tilde{\mathbf{Q}}_n$$

  where $\tilde{\mathbf{Q}}_k$ is given by computed $\tilde{\mathbf{v}}_k$, since $\mathbf{Q}$ is not formed explicitly

Outline

- Gaussian Elimination (without pivoting)
- Gaussian Elimination (with partial pivoting)

Gaussian Elimination and LU Factorization

- Gaussian elimination can be viewed as "triangular triangularization" of nonsingular $\mathbf{A} \in \mathbb{C}^{m \times m}$

$$\underbrace{\mathbf{L}_{m-1} \cdots \mathbf{L}_2 \mathbf{L}_1}_{\mathbf{L}^{-1}} \mathbf{A} = \mathbf{U}$$

  analogous to Householder QR factorization of matrix $\mathbf{A} \in \mathbb{C}^{m \times n}$

$$\underbrace{\mathbf{Q}_n \cdots \mathbf{Q}_2 \mathbf{Q}_1}_{\mathbf{Q}^*} \mathbf{A} = \mathbf{R}$$

- Example of LU factorization of 4×4 matrix $\mathbf{A}$

$$\underset{\mathbf{L}_1}{\rightarrow} \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix}}_{\mathbf{L}_1\mathbf{A}} \underset{\mathbf{L}_2\mathbf{A}}{\rightarrow} \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & 0 & \times & \times \\ & 0 & \times & \times \end{bmatrix}}_{\mathbf{L}_2\mathbf{L}_1\mathbf{A}} \underset{\mathbf{L}_3\mathbf{A}}{\rightarrow} \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & 0 & \times \end{bmatrix}}_{\mathbf{L}_3\mathbf{L}_2\mathbf{L}_1\mathbf{A}}$$

What is Matrices $\mathbf{L}_k$

- At step k, eliminate entries below a_{kk} : let x_k be kth column of $\mathbf{L}_{k-1}\cdots\mathbf{L}_2\mathbf{L}_1\mathbf{A}$,

$$x_k = [x_{1,k}, x_{2,k}, \cdots, x_{k,k}, x_{k+1,k}\cdots x_{m,k}]^T$$

$$\mathbf{L}_k x_k = [x_{1,k}, x_{2,k}, \cdots, x_{k,k}, 0\cdots 0]^T$$

- The multipliers $l_{jk} = x_{jk}/x_{kk}$ appear in $\mathbf{L}_k$

$$\begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & 0 & 1 & & \\ & 0 & -l_{k+1,k} & 1 & \\ & & \vdots & & \ddots \\ & 0 & \vdots & & \\ & & -l_{m,k} & & 1 \end{bmatrix}$$

- Let $l_k = [0, \cdots, 0, l_{k+1,k}, \cdots, l_{m,k}]^T$ and $e_k = [\underbrace{0, \cdots, 0}_{k-1}, 1, \cdots, 0]^T$, then $\mathbf{L}_k = \mathbf{I} - l_k e_k^*$

Forming L

- Luckily, the **L** matrix contains the multipliers $l_{jk} = x_{jk}/x_{kk}$

$$\mathbf{L} = \mathbf{L}_1^{-1}\mathbf{L}_2^{-1}\cdots\mathbf{L}_{m-1}^{-1} = \begin{bmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ l_{m1} & l_{m2} & \cdots & l_{m,m-1} & 1 \end{bmatrix}$$

  and is said to be a unit lower triangular matrix
- First, $\mathbf{L}_k^{-1} = \mathbf{I} + l_k\mathbf{e}_k^*$, because $\mathbf{e}_k^* k_k = 0$ and $(\mathbf{I} - l_k\mathbf{e}_k^*)(\mathbf{I} + l_k\mathbf{e}_k^*) = \mathbf{I} - l_k\mathbf{e}_k^* l_k\mathbf{e}_k^* = \mathbf{I}$
- Second, $L_1^{-1}L_2^{-1}\cdots L_{k+1}^{-1} = \mathbf{I} + \sum_{j=1}^{k+1} l_j\mathbf{e}_j^*$, since (prove by induction) $(\mathbf{I} + \sum_{j=1}^{k+1} l_j\mathbf{e}_j^*)(\mathbf{I} + l_{k+}\mathbf{e}_{k+1}^*) = \mathbf{I} + \sum_{j=1}^{k+1} l_j\mathbf{e}_j^* + \sum_{j=1}^{k} l_j(\mathbf{e}_j^* l_{k+1}\mathbf{e}_{k+1}^*)$ where $\mathbf{e}_j^* l_{k+1} = 0$ for $j < k+1$
- In other words, **L** is "union" of $\mathbf{L}_1^{-1}, \mathbf{L}_2^{-1}, \cdots, \mathbf{L}_{m-1}^{-1}$

Gaussian Elimination without Pivoting

- Factorize $\mathbf{A} \in \mathbb{C}^{m \times m}$ into $\mathbf{A} = \mathbf{LU}$

Gaussian elimination without pivoting:
$\mathbf{U} = \mathbf{A}, \mathbf{L} = \mathbf{I}$;
for $k = 1$ to $m-1$
  for $j = k+1$ to m
    $l_{jk} = u_{jk}/u_{kk}$
    $u_{j,k:m} = u_{j,k:m} - l_{jk} u_{k,k:m}$

- Flop count $\sim \sum_{k=1}^{m} 2(m-k)(m-k) \sim 2\sum_{k=1}^{m} k^2 \sim 2m^3/3$
- In actually, **L** often overwrites lower-triangular part of **A** and **U** overwrites upper-triangular part of **A**
- Question: What if u_{kk} is 0? Answer: The algorithm would break.

Partial Pivoting

- At step k, we divide by u_{kk} , which would break if u_{kk} is 0 (or close to 0), which can happen even if **A** is nonsingular

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ & x_{kk} & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \end{bmatrix} \rightarrow \begin{bmatrix} \times & \times & \times & \times & \times \\ & x_{kk} & \times & \times & \times \\ & 0 & \times & \times & \times \\ & 0 & \times & \times & \times \\ & 0 & \times & \times & \times \end{bmatrix}$$

- However, any nonzero entry in kth column below diagonal can also be used as pivot

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & x_{ik} & \times & \times & \times \\ & \times & \times & \times & \times \end{bmatrix} \rightarrow \begin{bmatrix} \times & \times & \times & \times & \times \\ & 0 & \times & \times & \times \\ & 0 & \times & \times & \times \\ & x_{ik} & \times & \times & \times \\ & 0 & \times & \times & \times \end{bmatrix}$$

  and we permute (interchange) row i with row k
- In general, we take nonzero entry with largest absolute value

More on Partial Pivoting

- kth step of Gaussian elimination of partial pivoting

$$\underbrace{\begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & x_{ik} & \times & \times \\ & \times & \times & \times \end{bmatrix}}_{\text{Pivotselection}} \xrightarrow{\mathbf{P}_k} \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ & x_{kk} & \times & \times \\ & \times & \times & \times \\ & \times & \times & \times \end{bmatrix}}_{\text{Rowinterchange}} \xrightarrow{\mathbf{L}_k} \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ & x_{kk} & \times & \times \\ & 0 & \times & \times \\ & 0 & \times & \times \end{bmatrix}}_{\text{Elimination}}$$

  and we interchange row i with row k
- In terms of matrices, it becomes $\underbrace{\mathbf{L}_{m-1}\mathbf{P}_{m-1}\cdots\mathbf{L}_2\mathbf{P}_2\mathbf{L}_1\mathbf{P}_1}_{\mathbf{L}^{-1}\mathbf{P}}\mathbf{A} = \mathbf{U}$
- $\mathbf{P} = \mathbf{P}_{m-1}\cdots\mathbf{P}_2\mathbf{P}_1$ and $\mathbf{L} = (\mathbf{L}'_{m-1}\cdots\mathbf{L}'_2\mathbf{L}'_1)^{-1}$, where $\mathbf{L}'_k = \mathbf{P}_{m-1}\cdots\mathbf{P}_{k+1}\mathbf{L}_k\mathbf{P}_{k+1}^{-1}\cdots\mathbf{P}_{m-1}^{-1}$
- It is easy to verify that $\mathbf{L}_{m-1}\mathbf{P}_{m-1}\cdots\mathbf{L}_2\mathbf{P}_2\mathbf{L}_1\mathbf{P}_1 = \mathbf{L}'_{m-1}\cdots\mathbf{L}'_2\mathbf{L}'_1(\mathbf{P}_{m-1}\cdots\mathbf{P}_2\mathbf{P}_1)$
- $\mathbf{L}'_k = \mathbf{I} - \mathbf{P}_{m-1}\cdots\mathbf{P}_{k+1}l_k\mathbf{e}_k^*$ and $\mathbf{L}$ is "union" of $(\mathbf{L}'_k)^{-1} \equiv \mathbf{I} + \mathbf{P}_{m-1}\cdots\mathbf{P}_{k+1}l_k\mathbf{e}_k^*$

Algorithm of Gaussian Elimination with Partial Pivoting

- Factorize $\mathbf{A} \in \mathbb{C}^{m \times m}$ into $\mathbf{PA} = \mathbf{LU}$

Gaussian elimination with partial pivoting:

$\mathbf{U} = \mathbf{A}, \mathbf{L} = \mathbf{I}, \mathbf{P} = \mathbf{I}$
for $k = 1$ to $m-1$
  $i \leftarrow \arg\max_{i \geq k} |u_{ik}|$
  $\mathbf{u}_{k,k:m} \leftrightarrow \mathbf{u}_{i,k:m}$
  $l_{k,1:k-1} \leftrightarrow l_{i,1:k-1}$
  $p_k \leftrightarrow p_i$
  for $j = k+1$ to m
    $l_{jk} = u_{jk}/u_{kk}$
    $u_{j,k:m} = u_{j,k:m} - l_{jk} u_{k,k:m}$

- Question: What if u_{kk} is 0?
- Flop count $\sim \sum_{k=1}^{m} 2(m-k)(m-k) \sim 2\sum_{k=1} k^2 \sim \frac{2}{3}m^3$, same as without pivoting

An Alternative Implementation

- In practice, L and U overwrite A and P is represented by a vector

Gaussian elimination with partial pivoting (alternative)
$\mathbf{p} = [1, 2, \cdots, m]$;
for $k = 1$ to $m-1$
  $i \leftarrow \arg\max_{i \geq k} |a_{ik}|$
  $\mathbf{a}_{k,1:m} \leftrightarrow \mathbf{a}_{i,1:m}$
  $p_k \leftrightarrow p_i$
  $a_{k+1:m,k} \leftarrow a_{k+1:m,k} / a_{k,k}$
  $\mathbf{A}_{k+1:m,k+1:m} \leftarrow \mathbf{A}_{k+1:m,k+1:m} - a_{k+1:m,k} * a_{k,k+1:m}$

- Using LU factorization to solve $Ax = b$:
  1. $\mathbf{PA} = \mathbf{LU}$; ($LU$ factorization with partial pivoting)
  2. $\mathbf{Ly} = \mathbf{Pb}$; (Forward substitution)
  3. $\mathbf{Ux} = \mathbf{y}$; (Back substitution)

Complete Pivoting

- More generally, we can use any nonzero entry
- In theory, any nonzero entry $(i,j), i \geq k, j \geq k$

$$\begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & \times & x_{ij} & \times \\ & \times & \times & \times \end{bmatrix} \rightarrow \begin{bmatrix} \times & \times & \times & \times \\ & \times & 0 & \times \\ & \times & x_{ij} & \times \\ & \times & 0 & \times \end{bmatrix}$$

  and we then permute row i with row k, column j with column k
- In matrix operations, it can be expressed as

$$\underbrace{\mathbf{L}_{m-1}\mathbf{P}_{m-1}\cdots\mathbf{L}_2\mathbf{P}_2\mathbf{L}_1\mathbf{P}_1}_{\mathbf{L}^{-1}\mathbf{P}}\mathbf{A}\underbrace{\mathbf{Q}_1\mathbf{Q}_2\cdots\mathbf{Q}_{m-1}}_{\mathbf{Q}} = \mathbf{U}$$

- Therefore, $\mathbf{PAQ} = \mathbf{LU}$ where $\mathbf{P} = \mathbf{P}_{m-1}\cdots\mathbf{P}_2\mathbf{P}_1$ and $\mathbf{L} = (\mathbf{L}'_{m-1}\cdots\mathbf{L}'_2\mathbf{L}'_1)^{-1}$
- However, complete pivoting is typically not used in practice because it increases cost in search of pivot and complexity of implementation

Stability of LU Factorization

Stability of LU without Pivoting

- For $\mathbf{A} = \mathbf{LU}$ computed without pivoting

$$\tilde{\mathbf{L}}\tilde{\mathbf{U}} = \mathbf{A} + \delta\mathbf{A}, \quad \frac{\|\delta\mathbf{A}\|}{\|\mathbf{L}\|\,\|\mathbf{U}\|} = O(\varepsilon_{machine})$$

- This is close to backward stability, except that we have $\|\mathbf{L}\|\,\|\mathbf{U}\|$ instead of $\|\mathbf{A}\|$ in the denominator
- Instability of Gaussian elimination can happen only if one or both of the factors $\mathbf{L}$ and $\mathbf{U}$ is large relative to size of $\mathbf{A}$
- Unfortunately, $\|\mathbf{L}\|$ and $\|\mathbf{U}\|$ can be arbitrarily large (even for well-conditioned $\mathbf{A}$), e.g.,

$$\mathbf{A} = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix} \begin{bmatrix} 10^{-20} & 1 \\ 0 & 1 - 10^{20} \end{bmatrix}$$

- Therefore, the algorithm is unstable

Stability of LU Partial Pivoting

- With pivoting, all entries of $\mathbf{L}$ are in $[-1,1]$, so $\|L\| = O(1)$
- To measure growth in $\mathbf{U}$, we introduce the growth factor $\rho = \frac{max_{i,j}|u_{ij}|}{max_{i,j}|a_{ij}|}$, and hence $\|\mathbf{U}\| = O(\rho\,\|\mathbf{A}\|)$
- We then have $\mathbf{PA} = \mathbf{LU}$

$$\tilde{\mathbf{L}}\tilde{\mathbf{U}} = \tilde{\mathbf{P}}\mathbf{A} + \delta\mathbf{A}, \quad \frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|} = O(\rho\varepsilon_{machine})$$

- If $|l_{ij}| < 1$ for each $i > j$(i.e., there is no tie for the pivoting), then $\tilde{\mathbf{P}} = \mathbf{P}$ for sufficiently small $\varepsilon_{machine}$
- If $\rho = O(1)$, then the algorithm is backward stable
- In fact, $\rho \leq 2^{m-1}$, so by definition ρ is a constant but can be very large

The Growth Factor

- ρ can indeed be as large as 2^{m-1}. Consider matrix

$$\begin{bmatrix} 1 & & & & 1 \\ -1 & 1 & & & 1 \\ -1 & -1 & 1 & & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & & & & 0 \\ -1 & 1 & & & 0 \\ -1 & -1 & 1 & & 0 \\ -1 & -1 & -1 & 1 & 0 \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & & & & 1 \\ & 1 & & & 2 \\ & & 1 & & 4 \\ & & & 1 & 8 \\ & & & & 16 \end{bmatrix}$$

  where growth factor $\rho = 16 = 2^{m-1}$
- $\rho = 2^{m-1}$ is as large as ρ can get. It can be catastrophic in practice
- Theoretically, Gaussian elimination with partial pivoting is backward stable according to formal definition
- However, in the worst case, Gaussian elimination with partial pivoting may be unstable for practical values of m

The Growth Factor in Practice

- Good news: Large ρ occurs only for very skewed matrices. Experimentally, one rarely see very large ρ
- Probability of large ρ decreases exponentially in ρ
- "If you pick a billion matrices at random, you will almost certainly not find one for which Gaussian elimination is unstable"
- In practice, ρ is no larger than $O(\sqrt{m})$. However, this behavior is not fully understood yet
- In conclusion,
  - Gaussian elimination with partial pivoting is backward stable
  - In theory, its error may grow exponentially in m
  - In practice, it is stable for matrices of practical interests

Cholesky Factorization

Hermitian Positive-Definite Matrices

- Symmetric matrix $\mathbf{A} \in \mathbb{R}^{m \times m}$ is symmetric positive definite (SPD) if $\mathbf{x}^T \mathbf{A}\mathbf{x} > 0$ for $\mathbf{x} \in \mathbb{R}^m \backslash \{0\}$
- Hermitian matrix $\mathbf{A} \in \mathbb{C}^{m \times m}$ is Hermitian positive definite (HPD) if $\mathbf{x}^* \mathbf{A}\mathbf{x} > 0$ for $\mathbf{x} \in \mathbb{C}^m \backslash \{0\}$
- If $\mathbf{A}$ is $m \times m$ HPD and $\mathbf{X} \in \mathbb{C}^{m \times n}$ has full column rank, then $\mathbf{X}^* \mathbf{A}\mathbf{X}$ is HPD
- Any principal submatrix (picking some rows and corresponding columns) of $\mathbf{A}$ is HPD and $a_{ii} > 0$
- HPD matrices have positive real eigenvalues and orthogonal eigenvectors
- Note: Most textbooks only talk about SPD or HPD matrices, but a positive-definite matrix does not need to be symmetric or Hermitian! A real matrix $\mathbf{A}$ is positive definite iff $\mathbf{A} + \mathbf{A}^T$ is SPD.

Cholesky Factorization

- Key idea: take advantage and preserve the properties of symmetry and positive-definiteness in factorization
- Eliminate below diagonal and to the right of diagonal

$$\mathbf{A} = \begin{bmatrix} a_{11} & \mathbf{w}^* \\ \mathbf{w} & \mathbf{K} \end{bmatrix} = \begin{bmatrix} \alpha & 0 \\ \frac{\mathbf{w}}{\alpha} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \alpha & \frac{\mathbf{w}^*}{\alpha} \\ 0 & \mathbf{K} - \frac{\mathbf{w}\mathbf{w}^*}{a_{11}} \end{bmatrix}$$

$$\begin{bmatrix} \alpha & 0 \\ \frac{\mathbf{w}}{\alpha} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \alpha & \frac{\mathbf{w}^*}{\alpha} \\ 0 & \mathbf{K} - \frac{\mathbf{w}\mathbf{w}^*}{a_{11}} \end{bmatrix} \begin{bmatrix} \alpha & \frac{\mathbf{w}^*}{\alpha} \\ 0 & \mathbf{I} \end{bmatrix} = \mathbf{R}_1^* \mathbf{A}_1 \mathbf{R}_1$$

where $\alpha = \sqrt{a_{11}}$, where $a_{11} > 0$

- $\mathbf{K} - \frac{\mathbf{w}\mathbf{w}^*}{a_{11}}$ is principal submatrix of HPD $\mathbf{A}_1 = \mathbf{R}_1^{-*}\mathbf{A}\mathbf{R}_1$ and therefore is HPD, with positive diagonal entries

Cholesky Factorization

- Apply recursively to obtain

$$\mathbf{A} = (\mathbf{R}_1^* \mathbf{R}_2^* \cdots \mathbf{R}_m^*)(\mathbf{R}_m \cdots \mathbf{R}_2 \mathbf{R}_1) = \mathbf{R}^* \mathbf{R}, \quad \mathbf{r}_{jj} > 0$$

  which is known as Cholesky factorization
- Question: Is $\mathbf{R}$ simply "union" of kth rows of $\mathbf{R}_k$ (or $\mathbf{R}^*$ "union" of kth columns of $\mathbf{R}_k^*$)? Yes. Hint: Write $\mathbf{R}_k^*$ in a form similar to $\mathbf{L}_k = \mathbf{I} + l_k \mathbf{e}_k^T$ in LU
- Existence and uniqueness: every HPD matrix has a unique Cholesky factorization
  - Exists because algorithm for Cholesky factorization always works for HPD matrices
  - Is unique since once $\alpha = \sqrt{a_{11}}$ is determined at each step, entire column $\frac{\mathbf{w}}{\alpha}$ is determined
  - Question: How to check whether a Hermitian matrix is positive definite? Answer: Run Cholesky factorization and it would succeeds iff the matrix is positive definite.

Algorithm of Cholesky Factorization

- Factorize Hermitian positive definite matrix $\mathbf{A} \in \mathbb{C}^{m \times m}$ into $\mathbf{A} = \mathbf{R}^* \mathbf{R}$

Algorithm: Cholesky factorization

$\mathbf{R} = \mathbf{A}$

for $k = 1$ to m

  for $j = k+1$ to m

    $r_{j,j:m} \leftarrow r_{j,j:m} - r_{k,j:m} \bar{r}_{kj} / r_{kk}$

  $r_{k,k:m} \leftarrow r_{k,k:m} / \sqrt{r_{kk}}$

- Operation count

$$\sum_{k=1}^{m} \sum_{j=k+1}^{m} 2(m-j) \sim 2 \sum_{k=1}^{m} \sum_{j=1}^{k} j \sim \sum_{k=1}^{m} k^2 \sim \frac{m^3}{3}$$

Stability

Theorem

The computed Cholesky factor $\tilde{\mathbf{R}}$ satisfies

$$\tilde{\mathbf{R}}^*\tilde{\mathbf{R}} = \mathbf{A} + \delta\mathbf{A}, \quad \frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|} = O(\varepsilon_{machine}),$$

i.e. Cholesky factorization is backward stable

- Forward errors in $\tilde{\mathbf{R}}$ is $\left\|\tilde{\mathbf{R}} - \mathbf{R}\right\| / \|\mathbf{R}\| = O(\kappa(\mathbf{A})\varepsilon_{machine})$, which may be large for ill-conditioned $\mathbf{A}$
- Solve $\mathbf{Ax} = \mathbf{b}$ for positive definite $\mathbf{A}$
  - Factorize $\mathbf{A} = \mathbf{R}^*\mathbf{R}$; Solve $\mathbf{R}^*\mathbf{y} = \mathbf{b}$; Solve $\mathbf{Rx} = \mathbf{y}$
  - Operation count is $\sim m^3/3$
  - Algorithm is backward stable:

LDL^* Factorization

- Cholesky factorization is sometimes given by $\mathbf{A} = \mathbf{LDL}^*$ where $\mathbf{D}$ is diagonal matrix and L is unit lower triangular matrix
- It avoids computing square roots
- Analogously, LU factorization can also be written as LDU, where $\mathbf{U}$ is unit upper triangular
- Question: How is $\mathbf{R}$ in $\mathbf{A} = \mathbf{R}^*\mathbf{R}$ related to the $\mathbf{L}$ and $\mathbf{U}$ factors of $\mathbf{A} = \mathbf{LU}$?
  - $\mathbf{U} = \mathbf{DL}^* = \sqrt{\mathbf{D}}\mathbf{R}$, where $\sqrt{\mathbf{D}} \equiv diag(\sqrt{d_{11}}, \sqrt{d_{22}}, \cdots, \sqrt{d_{mm}})$
- Hermitian indefinite systems can be factorized with $\mathbf{PAP}^T = \mathbf{LDL}^*$, but $\mathbf{D}$ is block diagonal with 1×1 and 2×2 blocks. Its cost is similar to Cholesky factorization and is about 50 of Gaussian elimination.

Software for Linear Algebra

Software for Linear Algebra

- LAPACK: Linear Algebra PACKage (www.netlib.org/lapack/lug)
  - Standard library for solving linear systems and eigenvalue problems
  - Successor of LINPACK (www.netlib.org/linpack) and EISPACK (www.netlib.org/eispack)
  - Depends on BLAS (Basic Linear Algebra Subprograms)
  - Parallel extensions include ScaLAPACK and PLAPACK
  - Note: Uses Fortran conventions for matrix arrangements

Software for Linear Algebra

- MATLAB
  - Factorization **A**: lu(**A**) and chol(**A**)
  - Solve $\mathbf{Ax} = \mathbf{b} : \mathbf{x} = \mathbf{A} \backslash \mathbf{b}$
    - Uses back/forward substitution for triangular matrices
    - Uses Cholesky factorization for positive-definite matrices
    - Uses LU factorization with column pivoting for nonsymmetric matrices
    - Uses Householder QR for least squares problems
    - Uses some special routines for matrices with special sparsity patterns
  - Uses LAPACK and other packages internally
- Serial and parallel solvers for sparse matrices (e.g., SuperLU, TAUCS)

Using LAPACK Routines in C Programs

- LAPACK was written in Fortran 77. Special attention is required when calling from C.
- Key differences between C and Fortran
  - Storage of matrices: column major (Fortran) versus row major (C/C++)
  - Argument passing for subroutines in C and Fortran: pass by reference (Fortran) and pass by value (C/C++)
- Smple example C code, example.c, for solving linear system using sgesv.
  - See class website for sample code.
  - To compile, issue command "cc -o example example.c -llapack -lblas"
- Hint: To find a function name, refer to LAPACK Users' Guide.
- To find out arguments for a given function, search on netlib.org

Numerical Linear Algebra
Module 5: Eigenvalues

Dr. Zahra Lakdawala

December 2, 2020

Outline

Eigenvalue problems

Eigenvalue and Eigenvectors

- Eigenvalue problem of $m \times m$ matrix $\mathbf{A}$ is

$$\mathbf{A}\mathbf{x} = \lambda \mathbf{x}$$

  with eigenvalues λ and eigenvectors $\mathbf{x}$ (nonzero)
- The set of all the eigenvalues of $\mathbf{A}$ is the spectrum of $\mathbf{A}$
- Eigenvalue are generally used where a matrix is to be compounded iteratively
- Eigenvalues are useful for algorithmic and physical reasons
  - Algorithmically, eigenvalue analysis can reduce a coupled system to a collection of scalar problems
  - Physically, eigenvalue analysis can be used to study resonance of musical instruments and stability of physical systems

Eigenvalue Decomposition

- Eigenvalue decomposition of $\mathbf{A}$ is

$$\mathbf{A} = \mathbf{X}\Lambda\mathbf{X}^{-1} \quad \text{or} \quad \mathbf{AX} = \mathbf{X}\Lambda$$

  with eigenvectors $\mathbf{x}$ i as columns of $\mathbf{X}$ and eigenvalues λ_i along diagonal of Λ. Alternatively,

$$\mathbf{A}\mathbf{x}_i = \lambda_i \mathbf{x}_i$$

- Eigenvalue decomposition is change of basis to "eigenvector coordinates"

$$\mathbf{A}\mathbf{x} = \mathbf{b} \rightarrow (\mathbf{X}\mathbf{b}^{-1}) = \Lambda(\mathbf{X}^{-1}\mathbf{x})$$

- Note that eigenvalue decomposition may not exist
- Question: How does eigenvalue decomposition differ from SVD?

Geometric Multiplicity

- Eigenvectors corresponding to a single eigenvalue λ form an eigenspace $E_\lambda \subseteq \mathbb{C}^{m\times m}$
- Eigenspace is invariant in that $\mathbf{A}E_\lambda \subseteq E_\lambda$
- Dimension of E_λ is the maximum number of linearly independent eigenvectors that can be found
- Geometric multiplicity of λ is dimension of E_λ , i.e., $\dim(\text{null}(\mathbf{A}-\lambda\mathbf{I}))$

Algebraic Multiplicity

- The characteristic polynomial of A is degree m polynomial

$$p_{\mathbf{A}}(z) = det(z\mathbf{I} - \mathbf{A}) = (z - \lambda_1)(z - \lambda_2)\cdots(z - \lambda_m)$$

  which is monic in that coefficient of z^m is 1

- λ is eigenvalue of $\mathbf{A}$ iff $p_{\mathbf{A}}(\lambda) = 0$
  - If λ is eigenvalue, then by definition, $\lambda\mathbf{x} - \mathbf{A}\mathbf{x} = (\lambda\mathbf{I} - \mathbf{A})\mathbf{x} = 0$, so $(\lambda\mathbf{I} - \mathbf{A})$ is singular and its determinant is 0
- Algebraic multiplicity of λ is its multiplicity as a root of $p_{\mathbf{A}}$
- Any matrix $\mathbf{A}$ has m eigenvalues, counted with algebraic multiplicity
- Question: What are the eigenvalues of a triangular matrix?
- Question: How are geometric multiplicity and algebraic multiplicity related?

Similarity Transformations

- The map $\mathbf{A} \rightarrow \mathbf{Y}^{-1}\mathbf{AY}$ is a similarity transformation of $\mathbf{A}$ for any nonsingular $\mathbf{Y} \in \mathbb{C}^{m \times m}$
- $\mathbf{A}$ and $\mathbf{B}$ are similar if there is similarity transformation $\mathbf{B} = \mathbf{Y}^{-1}\mathbf{AY}$

Theorem

If $\mathbf{Y}$ is nonsingular, then $\mathbf{A}$ and $\mathbf{Y}^{-1}\mathbf{AY}$ have the same characteristic polynomials, eigenvalues, and algebraic and geometric multiplicities.

1. For characteristic polynomial:

$$det(z\mathbf{I} - \mathbf{Y}^{-1}\mathbf{AY}) = det(\mathbf{Y}^{-1}(z\mathbf{I} - \mathbf{A})\mathbf{Y}) = det(z\mathbf{I} - \mathbf{A})$$

   so algebraic multiplicities remain the same

2. If $\mathbf{x} \in \mathbf{E}_\lambda$ for $\mathbf{A}$, then $\mathbf{Y}^{-1}\mathbf{x}$ is in eigenspace of $\mathbf{Y}^{-1}\mathbf{AY}$ corresponding to λ, and vice versa, so geometric multiplicities remain the same

Algebraic Multiplicity $\geq$ Geometric Multiplicity

- Let n be be geometric multiplicity of λ for $\mathbf{A}$. Let $\hat{\mathbf{V}} \in \mathbb{C}^{m \times n}$ constitute of orthonormal basis of the $\mathbf{E}_\lambda$
- Extend $\hat{\mathbf{V}}$ to unitary $\mathbf{V} = [\hat{\mathbf{V}}, \tilde{\mathbf{V}}] \in \mathbb{C}^{m \times m}$ and form

$$\mathbf{B} = \mathbf{V}^* \mathbf{A} \mathbf{V} = \begin{bmatrix} \hat{\mathbf{V}}^* \mathbf{A} \hat{\mathbf{V}} & \hat{\mathbf{V}}^* \mathbf{A} \tilde{\mathbf{V}} \\ \tilde{\mathbf{V}}^* \mathbf{A} \hat{\mathbf{V}} & \tilde{\mathbf{V}}^* \mathbf{A} \tilde{\mathbf{V}} \end{bmatrix} = \begin{bmatrix} \lambda \mathbf{I} & \mathbf{C} \\ 0 & \mathbf{D} \end{bmatrix}$$

- $\det(z\mathbf{I} - \mathbf{B}) = \det(z\mathbf{I} - \lambda \mathbf{I}\ ) \det(z\mathbf{I} - \lambda \mathbf{D}) = (z - \lambda)^n \det(z\mathbf{I} - \lambda \mathbf{D})$, so the algebraic multiplicity of λ as an eigenvalue of $\mathbf{B}$ is $\geq n$
- $\mathbf{A}$ and $\mathbf{B}$ are similar, so the algebraic multiplicity of λ as an eigenvalue of $\mathbf{A}$ is at least $\geq n$
- Examples:

$$\mathbf{A} = \begin{bmatrix} 2 & & \\ & 2 & \\ & & 2 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 2 & 1 & \\ & 2 & 1 \\ & & 2 \end{bmatrix}$$

  Their characteristic polynomial is $(z-2)^3$, so algebraic multiplicity of $\lambda = 2$ is 3. But geometric multiplicity of $\mathbf{A}$ is 3 and that of $\mathbf{B}$ is 1.

Defective and Diagonalizable Matrices

- An eigenvalue of a matrix is defective if its algebraic multiplicity $>$ its geometric multiplicity
- A matrix is defective if it has a defective eigenvalue. Otherwise, it is called nondefective.

Theorem

An $m \times m$ matrix $\mathbf{A}$ is nondefective iff it has an eigenvalue decomposition $\mathbf{A} = \mathbf{X}\Lambda\mathbf{X}^{-1}$.

Defective and Diagonalizable Matrices

- An eigenvalue of a matrix is defective if its algebraic multiplicity $>$ its geometric multiplicity
- A matrix is defective if it has a defective eigenvalue. Otherwise, it is called nondefective.

Theorem

An $m \times m$ matrix $\mathbf{A}$ is nondefective iff it has an eigenvalue decomposition $\mathbf{A} = \mathbf{X}\Lambda\mathbf{X}^{-1}$.

- ($\Leftarrow$) Λ is nondefective, and $\mathbf{A}$ is similar to Λ, so $\mathbf{A}$ is nondefective.
- ($\Rightarrow$) A nondefective matrix has m linearly independent eigenvectors. Take them as columns of $\mathbf{X}$ to obtain $\mathbf{A} = \mathbf{X}\Lambda\mathbf{X}^{-1}$.
- Nondefective matrices are therefore also said to be diagonalizable.

Determinant and Trace

- Determinant of $\mathbf{A}$ is $\det(\mathbf{A}) = \Pi_{j=1}^{m} \lambda_j$, because

$$det(\mathbf{A}) = (-1)^m det(-\mathbf{A}) = (-1)^m p_{\mathbf{A}}(0) = \Pi_{j=1}^{m} \lambda_j$$

- Trace of $\mathbf{A}$ is $\text{tr}(\mathbf{A}) = \sum_{j=1}^{m} \lambda_j$, since

$$p_{\mathbf{A}}(z) = det(z\mathbf{I} - \mathbf{A}) = z^m - \sum_{j=1}^{m} a_{jj} z^{m-1} + O(z^{m-2})$$

$$p_{\mathbf{A}}(z) = \Pi_{j=1}^{m}(z - \lambda_j) = z^m - \sum_{j=1}^{m} \lambda_j z^{m-1} + O(z^{m-2})$$

- Question: Are these results valid for defective or nondefective matrices?

Unitary Diagonalization

- A matrix $\mathbf{A}$ is unitarily diagonalizable if $\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^*$ for a unitary matrix $\mathbf{Q}$
- A hermitian matrix is unitarily diagonalizable, with real eigenvalues
- A matrix $\mathbf{A}$ is normal if $\mathbf{A}^*\mathbf{A} = \mathbf{A}\mathbf{A}^*$
  - Examples of normal matrices include hermitian matrices, skew hermitian matrices
  - hermitian $\Leftrightarrow$ matrix is normal and all eigenvalues are real
  - skew hermitian $\Leftrightarrow$ matrix is normal and all eigenvalues are imaginary
  - If $\mathbf{A}$ is both triangular and normal, then $\mathbf{A}$ is diagonal
- Unitarily diagonalizable $\Leftrightarrow$ normal
  - "$\Rightarrow$" is easy. Prove "$\Leftarrow$" by induction using Schur factorization next

Schur Factorization

- Schur factorization is $\mathbf{A} = \mathbf{QTQ}^*$, where $\mathbf{Q}$ is unitary and $\mathbf{T}$ is upper triangular

Theorem

Every square matrix $\mathbf{A}$ has a Schur factorization.

Proof by induction on dimension of $\mathbf{A}$. Case $m = 1$ is trivial. For $m \geq 2$, let $\mathbf{x}$ be any unit eigenvector of $\mathbf{A}$, with corresponding eigenvalue λ. Let $\mathbf{U}$ be unitary matrix with $\mathbf{x}$ as first column. Then

$$\mathbf{U}^*\mathbf{AU} = \begin{bmatrix} \lambda & \mathbf{w}^* \\ 0 & \mathbf{C} \end{bmatrix}$$

By induction hypothesis, there is a Schur factorization $\tilde{\mathbf{T}} = \mathbf{V}^*\mathbf{CV}$. Let

$$\mathbf{Q} = \mathbf{U}\begin{bmatrix} 1 & 0 \\ 0 & \mathbf{V} \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} \lambda & \mathbf{w}^*\mathbf{V} \\ 0 & \tilde{\mathbf{T}} \end{bmatrix}$$

and then $\mathbf{A} = \mathbf{QTQ}^*$

Eigenvalue Revealing Factorization

- Eigenvalue-revealing factorization of square matrix $\mathbf{A}$
  - Diagonalization $\mathbf{A} = \mathbf{X}\Lambda\mathbf{X}^{-1}$ (nondefective $\mathbf{A}$)
  - Unitary Diagonalization $\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^*$ (normal $\mathbf{A}$)
  - Unitary triangularization (Schur factorization) $\mathbf{A} = \mathbf{Q}\mathbf{T}\mathbf{Q}^*$ (any $\mathbf{A}$)
  - Jordan normal form $\mathbf{A} = \mathbf{X}\mathbf{J}\mathbf{X}$, where $\mathbf{J}$ block diagonal with

$$\mathbf{J}_i = \begin{bmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{bmatrix}$$

- In general, Schur factorization is used, because
  - Unitary matrices are involved, so algorithm tends to be more stable
  - If $\mathbf{A}$ is normal, then Schur form is diagonal

Eigenvalue algorithms

Eigenvalue Revealing Factorization

- Eigenvalue-revealing factorization of square matrix $\mathbf{A}$
  - Diagonalization $\mathbf{A} = \mathbf{X}\Lambda\mathbf{X}^{-1}$ (nondefective $\mathbf{A}$)
  - Unitary Diagonalization $\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^*$ (normal $\mathbf{A}$)
  - Unitary triangularization (Schur factorization) $\mathbf{A} = \mathbf{Q}\mathbf{T}\mathbf{Q}^*$ (any $\mathbf{A}$)
  - Jordan normal form $\mathbf{A} = \mathbf{X}\mathbf{J}\mathbf{X}$, where $\mathbf{J}$ block diagonal with

$$\mathbf{J}_i = \begin{bmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{bmatrix}$$

- In general, Schur factorization is used, because
  - Unitary matrices are involved, so algorithm tends to be more stable
  - If $\mathbf{A}$ is normal, then Schur form is diagonal

"Obvious" Algorithms

- Most obvious method is to find roots of characteristic polynomial $p_{\mathbf{A}}(\lambda)$, but it is very ill-conditioned
- Another idea is power iteration, using fact that

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

converge to an eigenvector corresponding to the largest eigenvalue of $\mathbf{A}$ in absolute value, but it may converge very slowly

"Obvious" Algorithms

- Most obvious method is to find roots of characteristic polynomial $p_{\mathbf{A}}(\lambda)$, but it is very ill-conditioned
- Another idea is power iteration, using fact that

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

  converge to an eigenvector corresponding to the largest eigenvalue of $\mathbf{A}$ in absolute value, but it may converge very slowly
- Instead, compute a eigenvalue-revealing factorization, such as Schur factorization

$$\mathbf{A} = \mathbf{Q}\mathbf{T}\mathbf{Q}^*$$

  by introducing zeros, using algorithms similar to QR factorization

A Fundamental Difficulty

- However, eigenvalue-revealing factorization cannot be done in finite number of steps:

  Any eigenvalue solver must be iterative

- To see this, consider a general polynomial of degree m

$$p(z) = z^m + a_{m-1}z^{m-1} + \cdots + a_1 z + a_0$$

  There is no closed-form expression for the roots of p: (Abel, 1842) In general, the roots of polynomial equations higher than fourth degree cannot be written in terms of a finite number of operations

A Fundamental Difficulty Cont'd

- However, the roots of $p_{\mathbf{A}}$ are the eigenvalues of the companion matrix

$$\mathbf{A} = \begin{bmatrix} 0 & & & & -a_0 \\ 1 & 0 & & & -a_1 \\ & 1 & \ddots & & \vdots \\ & & \ddots & 0 & -a_{m-2} \\ & & & 1 & -a_{m-1} \end{bmatrix}$$

- Therefore, in general, we cannot find the eigenvalues of a matrix in a finite number of steps
- In practice, however, there are algorithms that converge to desired precision in a few iterations

Schur Factorization and Diagonalization

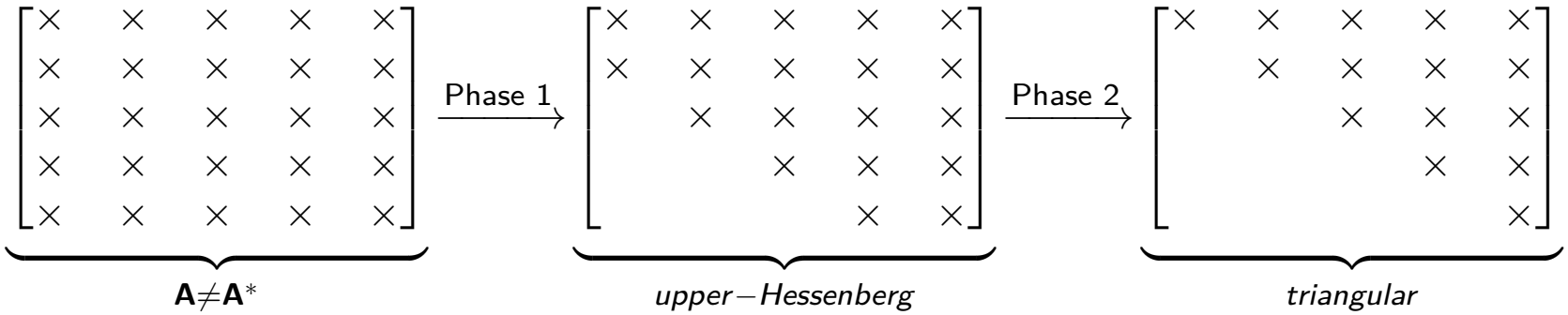
- Most eigenvalue algorithms compute Schur factorization $\mathbf{A} = \mathbf{QTQ}^*$ by transforming $\mathbf{A}$ with similarity transformations

$$\underbrace{\mathbf{Q}_j^* \cdots \mathbf{Q}_2^* \mathbf{Q}_1^*}_{\mathbf{Q}^*} \mathbf{A} \underbrace{\mathbf{Q}_1 \mathbf{Q}_2 \cdots \mathbf{Q}_j}_{\mathbf{Q}},$$

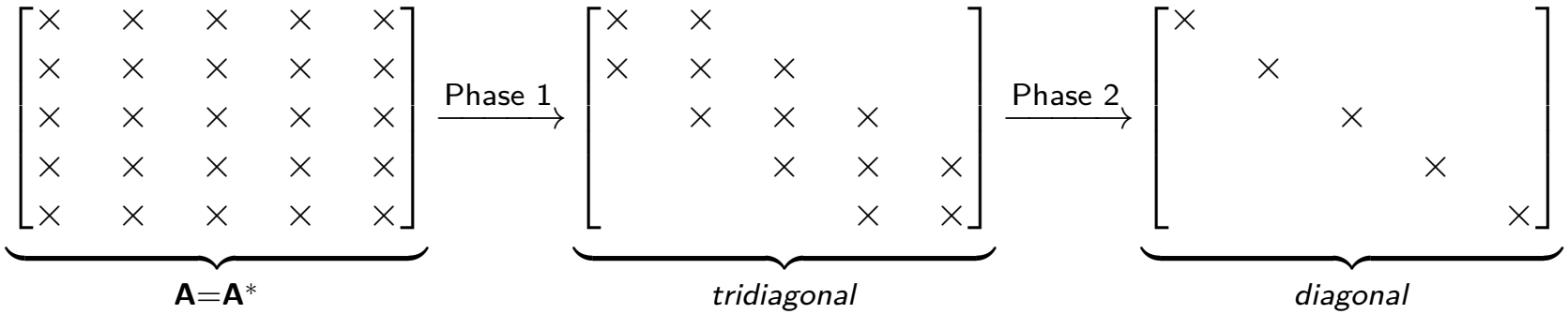
  where $\mathbf{Q}_i$ are unitary matrices, which converge to $\mathbf{T}$ as $j \to \infty$
- Note: Real matrices might need complex Schur forms and eigenvalues
- Question: For hermitian $\mathbf{A}$, what matrix will the sequence converge to?

Two Phases of Eigenvalue Computations

- General **A**: First convert to upper-Hessenberg form, then to upper triangular



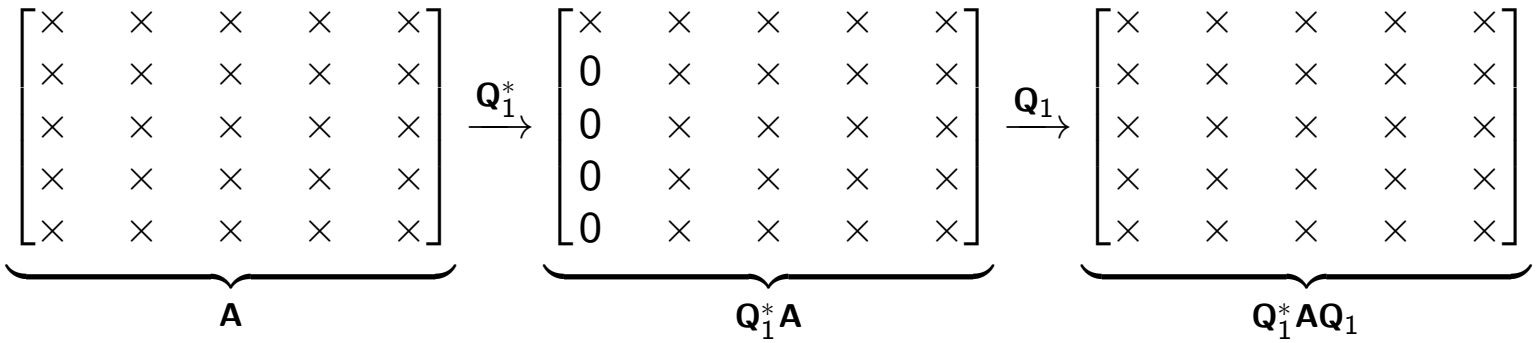
- Hermitian **A**: First convert to tridiagonal form, then to diagonal



- In general, phase 1 is direct and requires $O(m^3)$ flops, and phase 2 is iterative and requires $O(m)$ iterations, and $O(m^3)$ flops for non-Hermitian matrices and $O(m^2)$ flops for Hermitian matrices

Introducing Zeros by Similarity Transformations

- First attempt: Compute Schur factorization $\mathbf{A} = \mathbf{QTQ}^*$ by applying Householder reflectors from both left and right



- Unfortunately, the right multiplication destroys the zeros introduced by $\mathbf{Q}_1^*$ (has the effect of replacing each column by linear combination of all columns)
- This would not work because of Abel's theorem - no finite process can reveal the eigenvalues of $\mathbf{A}$
- However, the subdiagonal entries typically decrease in magnitude (even if it doesn't make it zero)

Hessenberg Form

The Hessenberg Form

- Second attempt: try to compute upper Hessenberg matrix $\mathbf{H}$ similar to $\mathbf{A}$:

$$\underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}}_{\mathbf{A}} \xrightarrow{\mathbf{Q}_1^*} \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix}}_{\mathbf{Q}_1^*\mathbf{A}} \xrightarrow{\mathbf{Q}_1} \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \end{bmatrix}}_{\mathbf{Q}_1^*\mathbf{A}\mathbf{Q}_1}$$

- The zeros introduced by $\mathbf{Q}_1^*\mathbf{A}$ were not destroyed this time!
- Continue with remaining columns would result in Hessenberg form:

$$\xrightarrow{\mathbf{Q}_2^*} \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & 0 & \times & \times & \times \\ & 0 & \times & \times & \times \end{bmatrix}}_{\mathbf{Q}_2^*\mathbf{Q}_1^*\mathbf{A}\mathbf{Q}_1} \xrightarrow{\mathbf{Q}_2} \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & \times & \times & \times \end{bmatrix}}_{\mathbf{Q}_2^*\mathbf{Q}_1^*\mathbf{A}\mathbf{Q}_1\mathbf{Q}_2}$$

The Hessenberg Form

- After $m-2$ steps, we obtain the Hessenberg form:

$$\underbrace{\mathbf{Q}_{m-2}^{*}\cdots\mathbf{Q}_{2}^{*}\mathbf{Q}_{1}^{*}}_{\mathbf{Q}^{*}}\mathbf{A}\underbrace{\mathbf{Q}_{1}\mathbf{Q}_{2}\cdots\mathbf{Q}_{m-2}}_{\mathbf{Q}}=\mathbf{H}=\begin{bmatrix}\times & \times & \times & \times & \times\\ \times & \times & \times & \times & \times\\ & \times & \times & \times & \times\\ & & \times & \times & \times\\ & & & \times & \times\end{bmatrix}$$

- For hermitian matrix $\mathbf{A}$, $\mathbf{H}$ is hermitian and hence is tridiagonal

Householder Reduction to Hessenberg

Algorithm: Householder Reduction to Hessenberg Form

for $k = 1$ to $m-2$

$\mathbf{x} = \mathbf{A}_{k+1:m,k}$

$\mathbf{v}_k = sign(\mathbf{x}_1)\|x\|_2 \mathbf{e}_1 + \mathbf{x}$

$\mathbf{v}_k = \mathbf{v}_k / \|\mathbf{v}_k\|_2$

$\mathbf{A}_{k+1:m,k:m} = \mathbf{A}_{k+1:m,k:m} - 2\mathbf{v}_k(\mathbf{v}_k^* \mathbf{A}_{k+1:m,k:m})$

$\mathbf{A}_{1:m,k+1:m} = \mathbf{A}_{1:m,k+1:m} - 2(\mathbf{A}_{1:m,k+1:m}\mathbf{v}_k)\mathbf{v}_k^*$

- Compare it to QR Factorization with Household Reflectors (Algorithm 10.1)
- Note: $\mathbf{Q}$ is never formed explicitly (as in Algorithm 10.1)
- Operation count

$$\sim \sum_{k=1}^{m-2} 4(m-k)^2 + 4m(m-k) \sim \frac{4m^3}{3} + 4m^3 - \frac{4m^3}{2} = \frac{10m^3}{3}$$

Reduction to Tridiagonal Form

- If **A** is hermitian, then

$$\underbrace{\mathbf{Q}_{m-2}^{*}\cdots\mathbf{Q}_{2}^{*}\mathbf{Q}_{1}^{*}}_{\mathbf{Q}^{*}}\mathbf{A}\underbrace{\mathbf{Q}_{1}\mathbf{Q}_{2}\cdots\mathbf{Q}_{m-2}}_{\mathbf{Q}}=\mathbf{H}=\begin{bmatrix}\times & \times & & & \\ \times & \times & \times & & \\ & \ddots & \ddots & \ddots & \\ & & \times & \times & \times \\ & & & \times & \times\end{bmatrix}$$

- For Hermitian A, operation count would be same as Householder QR: $\frac{4m^3}{3}$
  - First, taking advantage of sparsity, cost of applying right reflectors is also $4(m-k)^2$ instead of $4m(m-k)$, so cost is

$$\sim\sum_{k=1}^{m-2}8(m-k)^2\sim\frac{8m^3}{3}$$

  - Second, taking advantage of symmetry, cost is reduced by 50% to $\frac{4m^3}{3}$

Stability of Hessenberg Reduction

Theorem

Householder reduction to Hessenberg form is backward stable, in that

$$\tilde{\mathbf{Q}}\tilde{\mathbf{H}}\tilde{\mathbf{Q}}^* = \mathbf{A} + \delta\mathbf{A}, \qquad \frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|} = O(\varepsilon_{machine})$$

for some $\delta\mathbf{A} \in \mathbb{C}^{m\times m}$

Note: Similar to Householder QR, $\tilde{\mathbf{Q}}$ is exactly unitary based on some reflection vectors $\tilde{\mathbf{v}}_k$

Rayleigh Quotient, Inverse Iteration

Solving Eigenvalue Problems

- All eigenvalue solvers must be iterative
- Iterative algorithms have multiple facets:
  1. Basic idea behind the algorithms
  2. Convergence and techniques to speed-up convergence
  3. Efficiency of implementation
  4. Termination criteria
- We will focus on first two aspects

Simplification: Real Symmetric Matrices

- We will consider eigenvalue problems for real symmetric matrices, i.e. $\mathbf{A} = \mathbf{A}^{\mathbf{T}} \in \mathbb{R}^{m \times m}$, and $\mathbf{Ax} = \lambda \mathbf{x}$ for $\mathbf{x} \in \mathbb{R}^m$
- $\mathbf{A}$ has real eigenvalues $\lambda_1, \lambda_2, \cdots, \lambda_m$ and orthonormal eigenvectors $\mathbf{q}_1, \mathbf{q}_2, \cdots \mathbf{q}_m$, where $\|\mathbf{q}_j\| = 1$
- Eigenvalues are often also ordered in a particular way (e.g., ordered from large to small in magnitude)
- In addition, we focus on symmetric tridiagonal form
  - Why? Because phase 1 of two-phase algorithm reduces matrix into tridiagonal form

Rayleigh Quotient

- The Rayleigh quotient of $\mathbf{x} \in \mathbb{R}^m$ is the scalar

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

- For an eigenvector $\mathbf{x}$, its Rayleigh quotient is $r(\mathbf{x}) = \mathbf{x}^T \lambda \mathbf{x} / \mathbf{x}^T \mathbf{x} = \lambda$, the corresponding eigenvalue of $\mathbf{x}$
- For general $\mathbf{x}$, $r(\mathbf{x}) = \alpha$ that minimizes $\|\mathbf{A}\mathbf{x} - \alpha \mathbf{x}\|_2$.
- $\mathbf{x}$ is eigenvector of $\mathbf{A} \leftrightarrow \nabla r(\mathbf{x}) = \frac{2}{\mathbf{x}^T \mathbf{x}}(\mathbf{A}\mathbf{x} - r(\mathbf{x})\mathbf{x}) = 0$ with $x \neq 0$
- $r(\mathbf{x})$ is smooth and $\nabla r(\mathbf{q}_j) = 0$ for any j, and therefore is quadratically accurate:

$$r(\mathbf{x}) - r(\mathbf{q}_J) = O(\|x - q_J\|^2) \text{ as } x \to \mathbf{q}_J \text{ for some } J$$

Power Iteration

- Simple power iteration for largest eigenvalue

Algorithm: Power Iteration

$v^{(0)} =$ some unit-length vector
for $k = 1, 2, \cdots$
$\mathbf{w} = \mathbf{A}\mathbf{v}^{(k-1)}$
$\mathbf{v}^{(k)} = \mathbf{w} / \|\mathbf{w}\|$
$\lambda^{(k)} = r(\mathbf{v}^{(k)}) = (\mathbf{v}^{(k)})^T \mathbf{A}\mathbf{v}^{(k)}$

- Termination condition is omitted for simplicity

Convergence of Power Iteration

- Expand initial $v^{(0)}$ in orthonormal eigenvectors $\mathbf{q}_i$, and apply $\mathbf{A}^k$:

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

Theorem

If $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_m| \geq 0$ and $\mathbf{q}_1^T\mathbf{v}^{(0)} \neq 0$, this gives

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

as $k \to \infty$ where $\pm$ sign is chosen to be sign of $\mathbf{q}^T\mathbf{v}^{(k)}$

- It finds the largest eigenvalue (unless eigenvector is orthogonal to $\mathbf{v}^{(0)}$)
- Error reduces by only a constant factor ($\approx |\lambda_2/\lambda_1|$) each step, and very slowly especially when $|\lambda_2| \approx |\lambda_1|$

Limitations of the Power Iteration

- It can only find the eigenvector corresponding to the largest eigenvalue
- Convergence is linear, reducing the error only by a constant $\approx |\frac{\lambda_2}{\lambda_1}|$ at each iteration
- Quality of this algorithm depends on having a largest eigenvalue that is significantly larger than others
- Limited use, however powerful concept!!!

Inverse Iteration

- Apply power iteration on $(\mathbf{A}-\mu\mathbf{I})^{-1}$, with eigenvalues $\{(\lambda_j-\mu)-1\}$
- If $\mu \approx \lambda_J$ for some J, then $(\lambda_J-\mu)-1$ may be far larger than $(\lambda_j-\mu)^{-1}, j \neq J$, so power iteration may converge rapidly

Algorithm: Inverse Iteration

$v^{(0)}$ = some unit-length vector
for $k = 1, 2, \cdots$
  Solve $(\mathbf{A}-\mu\mathbf{I})\mathbf{w} = \mathbf{v}^{(k-1)}$ for $\mathbf{w}$
  $\mathbf{v}^{(k)} = \mathbf{w}/\|\mathbf{w}\|$
  $\lambda^{(k)} = r(\mathbf{v}^{(k)}) = (\mathbf{v}^{(k)})^T\mathbf{A}\mathbf{v}^{(k)}$

Convergence of Inverse Iteration

- Linear convergence (similar to the power iteration)
- Unlike power iteration, we can chose the eigenvector that will be found by suppling an estimate of μ of the corresponding eigenvalue.
- We can control the rate of linear convergence (chosing μ to be closer to the eigenvalue of **A**

Theorem

Suppose λ_J is the closest eigenvalue to μ and λ_K is the second closest, such that $|\mu - \lambda_J| < |\mu - \lambda_K| \leq |\mu - \lambda_j|$ for each $j \neq J$. The iterates converges to eigenvector $\mathbf{q}_J$ with

$$\left\|\mathbf{v}^{(k)} - (\pm \mathbf{q}_J)\right\| = O(|\frac{\mu - \lambda_J}{\mu - \lambda_K}|^k), |\lambda^{(k)} - \lambda_J| = O(|\frac{\mu - \lambda_J}{\mu - \lambda_K}|^{2k})$$

as $k \to \infty$.

- Standard method for determining eigenvector given eigenvalue (minus the Rayleigh quotient)

Rayleigh Quotient Iteration

- Parameter μ is constant in inverse iteration, but convergence is better for μ close to the eigenvalue
- Improvement: At each iteration, set μ to last computed Rayleigh quotient

Algorithm: Rayleigh Quotient Iteration

$\mathbf{v}^{(0)} =$ some unit-length vector
$\lambda^{(0)} = r(\mathbf{v}^{(0)}) = (\mathbf{v}^{(0)})^T \mathbf{A}\mathbf{v}^{(0)}$
for $k = 1, 2, \cdots$
  Solve $(\mathbf{A} - \lambda^{(k-1)}\mathbf{I})\mathbf{w} = \mathbf{v}^{(k-1)}$ for $\mathbf{w}$
  $\mathbf{v}^{(k)} = \mathbf{w}/\|w\|$
  $\lambda^{(k)} = r(\mathbf{v}^{(k)}) = (\mathbf{v}^{(k)})^T \mathbf{A}\mathbf{v}^{(k)}$

- Cost per iteration is linear for tridiagonal matrix

Convergence of Rayleigh Quotient Iteration

- Spectacular: Cubic convergence in Rayleigh quotient iteration

Theorem

Rayleigh Quotient Iteration converges to sn eigenvalue/eigenvector pair for all except a set of zero starting vectors. When it converges, the convergence is ultimately cubic in the sense that if λ_J is an eigenvalue of $\mathbf{A}$ and $\mathbf{v}^{(0)}$ is close to $\mathbf{q}_J$, then

$$\left\| v^{(k+1)} - (\pm \mathbf{q}_J) \right\| = O(\left\| \mathbf{v}^{(k)} - (\pm \mathbf{q}_J) \right\|^3)$$

and

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

as $k \to \infty$.

- In other words, each iteration triples number of digits of accuracy
- Rayleigh quotient is great in finding largest (or smallest) eigenvalue and its corresponding eigenvector. What if we want to find all eigenvalues?

Operation Counts

In Rayleigh quotient iteration:

- if $\mathbf{A} \in \mathbb{R}^{m \times m}$ is full matrix, then solving $(\mathbf{A} - \mu \mathbf{I})\mathbf{w} = \mathbf{v}^{(k-1)}$ may take $O(m^3)$ flops per step
- if $\mathbf{A} \in \mathbb{R}^{m \times m}$ is upper Hessenberg, then each step takes $O(m^2)$ flops
- if $\mathbf{A} \in \mathbb{R}^{m \times m}$ is tridiagonal, then each step takes $O(m)$ flops

QR

QR Algorithm

- Most basic version of QR algorithm is remarkably simple:

> Algorithm: "Pure" QR Algorithm
>
> $\mathbf{A}^{(0)} = \mathbf{A}$
> for $k = 1, 2, \cdots$
> $\mathbf{Q}^{(k)}\mathbf{R}^{(k)} = \mathbf{A}^{(k-1)}$
> $\mathbf{A}^{(k)} = \mathbf{R}^{(k)}\mathbf{Q}^{(k)}$

- With some suitable assumptions, $\mathbf{A}^{(k)}$ converge to Schur form of $\mathbf{A}$ (diagonal if $\mathbf{A}$ is symmetric)
- Similarity transformation of $\mathbf{A}$:

$$\mathbf{A}^{(k)} = \mathbf{R}^{(k)}\mathbf{Q}^{(k)} = (\mathbf{Q}^{(k)})^T\mathbf{A}^{(k-1)}\mathbf{Q}^{(k)}$$

- But why does it work?

Unnormalized Simultaneous Iteration

- To understand QR algorithm, first consider simple algorithm
- Simultaneous iteration is power iteration applied to several vectors
- Start with linearly independent $\mathbf{v}_1^{(0)}, \cdots, \mathbf{v}_n^{(0)}$
- We know from power iteration that $\mathbf{A}^k\mathbf{v}_1$ converge to $\mathbf{q}_1$
- With some assumptions, the space $\langle \mathbf{A}^k\mathbf{v}_1^{(0)}, \cdots, \mathbf{A}^k\mathbf{v}_n^{(0)} \rangle$ should converge to $\langle \mathbf{q}_1, \cdots, \mathbf{q}_n \rangle$
- Notation: Define initial matrix $\mathbf{V}^{(0)}$ and matrix $\mathbf{V}^{(k)}$ at step k:

$$\mathbf{V}^{(0)} = [\mathbf{v}_1^{(0)} | \cdots | \mathbf{v}_n^{(0)}], \quad \mathbf{V}^{(k)} = \mathbf{A}^k\mathbf{V}^{(0)} = [\mathbf{v}_1^{(k)} | \cdots \mathbf{v}_n^{(k)}]$$

Unnormalized Simultaneous Iteration

- Define orthogonal basis for column space of $\mathbf{V}^{(k)}$ by reduced QR factorization $\hat{\mathbf{Q}}^{(k)}\hat{\mathbf{R}}^{(k)} = \mathbf{V}^{(k)}$
- We assume that
  1. leading $n+1$ eigenvalues are distinct, and
  2. all leading principal submatrices of $\hat{\mathbf{Q}}^T\mathbf{V}^{(0)}$ are nonsingular where $\hat{\mathbf{Q}} = [\mathbf{q}_1|\cdots|\mathbf{q}_n]$
- We then have columns of $\hat{\mathbf{Q}}^{(k)}$ converge to eigenvectors of $\mathbf{A}$:

$$\left\|\mathbf{q}_j^{(k)} - (\pm\mathbf{q}_j)\right\| = O(C^k),$$

  where $c = max_{1\leq k\leq n}|\lambda_{k+1}|/|\lambda_k|$
- Proof idea: Show that subspace of any leading j columns of $\mathbf{V}^{(k)} = \mathbf{A}^k\mathbf{V}^{(0)}$ converges to subspace of first j eigenvectors of $\mathbf{A}$, so does the subspace of any leading j columns of $\hat{\mathbf{Q}}^{(k)}$.

The Idea

- We know that other eigenvectors are orthogonal to the dominant one
- so we can use the power method, and force that the second vector is orthogonal to the first one
- this way we guarantee that they will converge to two different eigenvectors
- we can do this for many vectors, not just two
- this is called "Simaltaneous Iteration"

Simultaneous/Orthogonal Iteration

- Matrices $\mathbf{V}^{(k)} = \mathbf{A}^k \mathbf{V}^{(0)}$ are highly ill-conditioned
- Orthonormalize at each step rather than at the end

Algorithm: Simultaneous Iteration

Pick $\hat{\mathbf{Q}}^{(0)} \in \mathbb{R}^{m \times n}$
for $k = 1, 2, \cdots$
$\quad \mathbf{Z}^{(k)} = \mathbf{A}\hat{\mathbf{Q}}^{(k-1)}$;
$\quad \hat{\mathbf{Q}}^{(k)}\hat{\mathbf{R}}^{(k)} = \mathbf{Z}^{(k)}$

- Column spaces of $\hat{\mathbf{Q}}^{(k)}$ and $\mathbf{Z}^{(k)}$ are both equal to column space of $\mathbf{A}^k\hat{\mathbf{Q}}^{(0)}$, therefore same convergence as before

Simultaneous Iteration $\Leftrightarrow$ QR Algorithm

Simultaneous Iteration

Pick $\hat{\mathbf{Q}}^{(0)} \in \mathbb{R}^{m \times n}$

for $k = 1, 2, \cdots$

$\mathbf{Z} = \mathbf{A}\hat{\mathbf{Q}}^{(k-1)}$

$\hat{\mathbf{Q}}^{(k)}\hat{\mathbf{R}}^{(k)} = \mathbf{Z}$

"Pure" QR Algorithm

$\mathbf{A}^{(0)} = \mathbf{A}$

for $k = 1, 2, \cdots$

$\mathbf{Q}^{(k)}\mathbf{R}^{(k)} = \mathbf{A}^{(k-1)}$

$\mathbf{A}^{(k)} = \mathbf{R}^{(k)}\mathbf{Q}^{(k)}$

- QR algorithm is equivalent to simultaneous iteration with $\hat{\mathbf{Q}}^{(0)} = \mathbf{I}$
- Since the matrices are now square, get rid of the hats. Replace $\hat{\mathbf{R}}^{(k)}$ by $\mathbf{R}^{(k)}$ and $\hat{\mathbf{Q}}^{(k)}$ by $\underline{\mathbf{Q}}^{(k)}$ (underline to differentiate between Simultaneous and QR algorithm).

Simultaneous Iteration $\Leftrightarrow$ QR Algorithm

- Further, we introduce a new statement $\mathbf{A}^{(k)} = (\underline{\mathbf{Q}}^{(k)})^T \mathbf{A} \underline{\mathbf{Q}}^{(k)}$ in simultaneous iteration

Simultaneous Iteration

Pick $\hat{\mathbf{Q}}^{(0)} \in \mathbb{R}^{m \times n}$

for $k = 1, 2, \cdots$

$\mathbf{Z} = \mathbf{A}\underline{\mathbf{Q}}^{(k-1)}$

$\mathbf{Z} = \underline{\mathbf{Q}}^{(k)}\mathbf{R}^{(k)}$

$\mathbf{A}^{(k)} = (\underline{\mathbf{Q}}^{(k)})^T \mathbf{A} \underline{\mathbf{Q}}^{(k)}$

"Pure" QR Algorithm

$\mathbf{A}^{(0)} = \mathbf{A}$

for $k = 1, 2, \cdots$

$\mathbf{A}^{(k-1)} = \mathbf{Q}^{(k)}\mathbf{R}^{(k)}$

$\mathbf{A}^{(k)} = \mathbf{R}^{(k)}\mathbf{Q}^{(k)}$

$\underline{\mathbf{Q}}^{(k)} = \mathbf{Q}^{(1)}\mathbf{Q}^{(2)} \ldots \mathbf{Q}^{(k)}$

- Let's look at the sequence of $\mathbf{R}^{(k)}$: $\mathbf{R}^k = (\mathbf{Q}^{(k)})^T \mathbf{Z}^{(k)} = (\mathbf{Q}^{(k)})^T \mathbf{A}\mathbf{Q}^{(k-1)}$
- if $\mathbf{Q}_k$ converges to some $\mathbf{Q}$ then $\mathbf{Q}^T\mathbf{A}\mathbf{Q} = \mathbf{R}$ is upper triangular
- This is a Schur Decomposition of $\mathbf{A}$
- Thus, the eigenvalues of $\mathbf{A}$ are located on the main diagonal of $\mathbf{R}$
- And the columns of $\mathbf{Q}$ are the eigenvectors

Simultaneous Iteration $\Leftrightarrow$ QR Algorithm

- $\underline{\mathbf{Q}}^{(k)} = \mathbf{Q}^{(1)}\mathbf{Q}^{(2)}\dots\mathbf{Q}^{(k)}$. Let $\underline{\mathbf{R}}^{(k)} = \mathbf{R}^{(k)}\mathbf{R}^{(k-1)}\dots\mathbf{R}^{(1)}$

Theorem

Both schemes generate QR factorization of k-th power of A:

$$\mathbf{A}^k = \underline{\mathbf{Q}}^{(k)}\underline{\mathbf{R}}^{(k)}$$

and projection

$$\mathbf{A}^{(k)} = (\underline{\mathbf{Q}}^{(k)})^T\mathbf{A}\underline{\mathbf{Q}}^{(k)}$$

Proof by induction. For $k = 0$ it is trivial for both algorithms.
For $k \geq 1$ with simultaneous iteration, $A^{(k)}$ is given by definition, and

$$\mathbf{A}^k = \underline{\mathbf{Q}}^{(k-1)}\underline{\mathbf{R}}^{(k-1)} = \underline{\mathbf{Q}}^{(k)}\mathbf{R}^{(k)}\underline{\mathbf{R}}^{(k-1)} = \underline{\mathbf{Q}}^{(k)}\underline{\mathbf{R}}^{(k)}$$

For $k \geq 1$ with QR algorithm,

$$\mathbf{A}^k = \underline{\mathbf{Q}}^{(k-1)}\underline{\mathbf{R}}^{(k-1)} = \underline{\mathbf{Q}}^{(k-1)}\mathbf{A}^{(k-1)}\underline{\mathbf{R}}^{(k-1)} = \underline{\mathbf{Q}}^{(k)}\underline{\mathbf{R}}^{(k)}$$

and

$$\mathbf{A}^{(k)} = (\mathbf{Q}^{(k)})^T\mathbf{A}^{(k-1)}\mathbf{Q}^{(k)} = (\underline{\mathbf{Q}}^{(k)})^T\mathbf{A}\underline{\mathbf{Q}}^{(k)}$$

Convergence of the unshifted QR Algorithm

- Since $\underline{\mathbf{Q}}^{(k)} = \hat{\mathbf{Q}}^{(k)}$ in simultaneous iteration, column vectors of $\underline{\mathbf{Q}}^{(k)}$ converge linearly to eigenvectors if $\mathbf{A}$ has distinct eigenvalues
- $\mathbf{A}^{(k)} = (\underline{\mathbf{Q}}^{(k)})^T \mathbf{A} \underline{\mathbf{Q}}^{(k)}$, diagonal entries of $\mathbf{A}^{(k)}$ are Rayleigh quotients of column vectors of $\underline{\mathbf{Q}}^{(k)}$, so they converge linearly to eigenvalues of $\mathbf{A}$
- Off-diagonal entries of $\mathbf{A}^{(k)}$ converge to zeros, as they are generalized Rayleigh quotients involving approximations of distinct eigenvectors
- Overall, $\mathbf{A} = \underline{\mathbf{Q}}^{(k)} \mathbf{A}^{(k)} (\underline{\mathbf{Q}}^{(k)})^T$. For a symmetric matrix, it converges to eigenvalue decomposition of $\mathbf{A}$