Mathematics Methods for Computer Science

Instructor: Xubo Yang

SJTU-SE DALAB

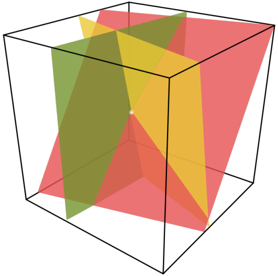
Lecture

Linear Systems and LU

Linear System

$$A\vec{x} = \vec{b}$$ (线性系统的关键方程)

$$A \in \mathbb{R}^{m\times n},\ \vec{x} \in \mathbb{R}^{n},\ \vec{b} \in \mathbb{R}^{m}$$



Case 1: Solvable

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

"Completely Determined"

Case 2: No Solution

$$\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

"OverDetermined"

过定，即无解

Case 3: Infinitely Many Solutions

$$\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \end{pmatrix}$$

"UnderDetermined"

无穷多解

No Other Cases

Proposition

If $A\vec{x} = \vec{b}$ has two distinct solutions $\vec{x}_0$ and $\vec{x}_1$, it has infinitely many solutions.

这个是一定正确的，因为对于x_0与x_1,取任意0<c<1,则有无穷多的新的x=c * x_0 + (1 - c) * x_1,满足线性方程。

Common Misconception

Solvability can depend on $\vec{b}$!

$$\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$$

Dependence on Shape of Matrix A

tall指的是m>n

Proposition

Tall matrices admit unsolvable right hand sides.

wide指的是m<n

Proposition

Wide matrices admit right hand sides with infinite numbers of solutions.

or

No wide matrix system admits a unique solution.

wide一般有无穷多解，而tall则存在有唯一解、无解等多种可能。

For Now

All matrices will be:

- Square: $A \in \mathbb{R}^{n \times n}$,
- Invertible: nonsingular, i.e. $A\vec{x} = \vec{b}$ is solvable for any $\vec{b}$

invertible：可逆的

Inverting Matrices

Do not compute A^{-1} if you do not need it.

- Not the same as solving $A\vec{x} = \vec{b}$
- Can be slow and poorly conditioned

Example

$$\begin{aligned} y - z &= -1 \\ 3x - y + z &= 4 \\ x + y - 2z &= -3 \end{aligned} \iff \left(\begin{array}{ccc|c} 0 & 1 & -1 & -1 \\ 3 & -1 & 1 & 4 \\ 1 & 1 & -2 & -3 \end{array}\right)$$

- Permute rows 换行
- Row scaling 对某一行成一个系数
- Forward/back substitution 进行两行之间的线性组合

Row Operations: Permutation

这个映射意思是说，将m维矩阵的每一行的索引对应到一个新的索引，从而实现换行操作，如：{1,2,3}->{3,1,2}

$$\sigma : \{1, \ldots, m\} \to \{1, \ldots, m\}$$

$$P_\sigma \equiv \begin{pmatrix} - & \vec{e}_{\sigma(1)}^{\top} & - \\ - & \vec{e}_{\sigma(2)}^{\top} & - \\ & \cdots & \\ - & \vec{e}_{\sigma(m)}^{\top} & - \end{pmatrix}$$

Row Operations: Row Scaling

$$S_a \equiv \begin{pmatrix} a_1 & 0 & 0 & \cdots \\ 0 & a_2 & 0 & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_m \end{pmatrix}$$

Row Operations: Elimination

(很常用)

"Scale row k by constant c and add result to row l."

$$E \equiv \left(I + c\vec{e}_l \vec{e}_k^{\,T}\right)$$

e_l与e_k均为列向量，作用就是取出原来矩阵中的某一行。

Solving via Elimination Matrices

$$\left(\begin{array}{ccc|c} 0 & 1 & -1 & -1 \\ 3 & -1 & 1 & 4 \\ 1 & 1 & -2 & -3 \end{array}\right)$$

Reverse order!

Introducing Gaussian Elimination

Big idea:

General strategy to solve linear systems via row operations.

Elimination Matrix Interpretation

$$A\vec{x} = \vec{b}$$

$$E_1 A\vec{x} = E_1\vec{b}$$

$$E_2 E_1 A\vec{x} = E_2 E_1\vec{b}$$

$$\vdots$$

$$\underbrace{E_k \cdots E_2 E_1 A}_{I_{n\times n}}\vec{x} = \underbrace{E_k \cdots E_2 E_1}_{A^{-1}}\vec{b}$$

Shape of Systems

$$(A|\vec{b}) \equiv \left(\begin{array}{cccc|c} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{array}\right)$$

增广矩阵

Pivot

$$\left(\begin{array}{cccc|c} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{array}\right)$$

Row Scaling

$$\left(\begin{array}{cccc|c} 1 & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{array}\right)$$

Row Scaling

$$\left(\begin{array}{cccc|c} 1 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{array}\right)$$

Forward Substitution

上面的行乘以系数，然后改变下面的行的数值

$$\left(\begin{array}{cccc|c} 1 & \times & \times & \times & \times \\ 0 & 1 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \end{array}\right)$$

Upper Triangular Form

对矩阵A从上到下顺序执行forward substitution之后即可以得到上三角矩阵

$$\left(\begin{array}{cccc|c} 1 & \times & \times & \times & \times \\ 0 & 1 & \times & \times & \times \\ 0 & 0 & 1 & \times & \times \\ 0 & 0 & 0 & 1 & \times \end{array}\right)$$

Back Substitution

与forward方向相反，从下至上

$$\left(\begin{array}{cccc|c} 1 & \times & \times & 0 & \times \\ 0 & 1 & \times & 0 & \times \\ 0 & 0 & 1 & 0 & \times \\ 0 & 0 & 0 & 1 & \times \end{array}\right)$$

Back Substitution

$$\left(\begin{array}{cccc|c} 1 & \times & 0 & 0 & \times \\ 0 & 1 & 0 & 0 & \times \\ 0 & 0 & 1 & 0 & \times \\ 0 & 0 & 0 & 1 & \times \end{array}\right)$$

Steps of Gaussian Elimination

- Forward substitution: For each row $i = 1, 2, \ldots, m$
  - Scale row to get pivot 1
  - For each $j > i$, subtract multiple of row i from row j to zero out pivot column
- Backward substitution: For each row $i = m, m-1, \ldots, 1$
  - For each $j < i$, zero out rest of column

Total Running Time

$$O(n^3)$$

n^2 * n,第一个n^2表示元素个数，第二个n表示进行消去时每一个元素都被处理了n次(因为每次进行行之间的线性组合都会直接影响到此行的所有元素。)

Problem

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

直接交换两行

Even Worse

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

Pivoting

Pivoting

Permuting rows and/or columns to avoid dividing by small numbers or zero.

- Partial pivoting
- Full pivoting

$$\begin{pmatrix} 1 & 10 & -10 \\ 0 & 0.1 & 9 \\ 0 & 4 & 6.2 \end{pmatrix}$$

Elimination Matrix Interpretation

$$A\vec{x}_1 = \vec{b}_1$$
$$A\vec{x}_2 = \vec{b}_2$$
$$\vdots$$

Can we restructure A to make this more efficient?

Does each solve take $O(n^3)$ time?

Observation

Steps of Gaussian elimination depend only on structure of A.

Avoid repeating identical arithmetic on A?

Another Clue: Upper Triangular Systems

$$\left(\begin{array}{cccc|c} 1 & \times & \times & \times & \times \\ 0 & 1 & \times & \times & \times \\ 0 & 0 & 1 & \times & \times \\ 0 & 0 & 0 & 1 & \times \end{array}\right)$$

After Back Substitution

这里说的是，对于一个上三角矩阵，可以再三角矩阵的基础上使用back方式进行进一步的优化，从而减少没必要进行的操作的次数(如x-0)

$$\left(\begin{array}{cccc|c} 1 & \times & \times & 0 & \times \\ 0 & 1 & \times & 0 & \times \\ 0 & 0 & 1 & 0 & \times \\ 0 & 0 & 0 & 1 & \times \end{array}\right)$$

No need to subtract the 0's explicitly!

$O(n)$ time

Next Pivot: Same Observation

$$\left(\begin{array}{cccc|c} 1 & \times & 0 & 0 & \times \\ 0 & 1 & 0 & 0 & \times \\ 0 & 0 & 1 & 0 & \times \\ 0 & 0 & 0 & 1 & \times \end{array}\right)$$

1+2+3+。。。+n

Observation

Triangular systems can be solved in $O(n^2)$ time.

No need to subtract the 0's explicitly!

$O(n)$ time

Upper Triangular Part of A

$$A\vec{x} = \vec{b}$$

$$\vdots$$

$$M_k \cdots M_1 A\vec{x} = M_k \cdots M_1 \vec{b}$$

Define:

$$U \equiv M_k \cdots M_1 A$$

这里的M_i指的就是前面提到的转换矩阵，并且对于进行行之间线性组合的矩阵M，其本身就是下三角矩阵

Lower Triangular Part

$$
\begin{aligned}
U &= M_k \cdots M_1 A \\
\Rightarrow A &= (M_1^{-1} \cdots M_k^{-1}) U \\
&\equiv LU
\end{aligned}
$$

Why Is L Triangular?

E本身就是下三角矩阵，S也一定是对角阵，所以这些矩阵的乘积仍然为下三角矩阵，即为L。

$$S \equiv diag(a_1, a_2, \cdots)$$

$$E \equiv I + c\vec{e_l}\vec{e_l}^T$$

Proposition

The product of triangular matrices is triangular.

(可以考虑使用三角矩阵的定义进行证明，具体过程看教材)

Solving Systems Using LU

$$A\vec{x} = \vec{b}$$
$$\Rightarrow LU\vec{x} = \vec{b}$$

1. Solve $L\vec{y} = \vec{b}$ using forward substitution.
2. Solve $U\vec{x} = \vec{y}$ using backward substitution.

$O(n^2)$ (given LU factorization)

Solving Systems Using LU

For example:

$$A\vec{x} = \vec{b} \qquad \text{set} \quad \vec{y} = U\vec{x}$$

$$\Rightarrow LU\vec{x} = \vec{b}$$

$$\Rightarrow L\vec{y} = \vec{b} \qquad \Rightarrow \vec{y} = L^{-1}\vec{b}$$

$$\Rightarrow U\vec{x} = \vec{y} \qquad \Rightarrow \vec{x} = U^{-1}\vec{y}$$

It's easier to get the inverse matrix of L/U than A.

Applications of LU

1. Machine learning.
2. Linear regression.
3. Image processing.
4. Computer graphics.
5. ...

Any linear equations solving process.

LU: Compact Storage

$$\begin{pmatrix} U & U & U & U \\ L & U & U & U \\ L & L & U & U \\ L & L & L & U \\ L & L & L & L \end{pmatrix}$$

Assumption: Diagonal elements of L are 1.

Warning: Do not multiply this matrix!

Computing LU Factorization

Small modification of forward substitution step to keep track of L.[1]

[1] See textbook for pseudocode.

Question

Does every A admit a factorization

$$A = LU?$$

不一定。因为对于A中的某些元素，在进行矩阵变换之后可能会出现精度问题。

Recall: Pivoting

Pivoting

Permuting rows and/or columns to avoid dividing by small numbers or zero.

- Partial pivoting
- Full pivoting

$$\begin{pmatrix} 1 & 10 & -10 \\ 0 & 0.1 & 9 \\ 0 & 4 & 6.2 \end{pmatrix}$$

Pivoting by Swapping Columns

recall：对于矩阵A，对行的变化M表示为MA，对列的变化表示为AN。

$$\underbrace{(E_k \cdots E_1)}_{\text{elimination}} \cdot A \cdot \underbrace{(P_1 \cdots P_\ell)}_{\text{permutations}} \cdot \underbrace{(P_\ell^\top \cdots P_1^\top)}_{\text{inv. permutations}} \vec{x}$$

$$= (E_k \cdots E_1)\,\vec{b}$$

$$\Downarrow$$

$$A = LUP$$