STA 141C - Big Data & High Performance Statistical Computing	**Spring 2022**

Week 3-2: Gaussian elimination and pivoting

Lecturer: Bo Y.-C. Ning *April 14, 2022*

Announcement

- HW1 question 5

Last time

- LU decomposition
- Solving linear system with upper and lower triangulars

Today

- Gaussian elimination
- Pivoting

1 Gaussian elimination and LU decomposition

Recall that we want to solve the linear equation

$$Ax = b,$$

where A is a dense matrix (not necessarily symmetric), $A \in \mathbb{R}^{n\times n}$, $x, b \in \mathbb{R}^n$.

The idea is to use a series of elementary operations called *Gaussian elimination* (proposed by Carl Friedrich Gauss in 1800s) to turn A into a triangular system and then apply forward and backward substitutions to solve x.

1.1 Gaussian elimination

Introducing the elementary operator matrix $E_{jk}(c)$, an identity matrix with 0 in the position (j, k) replaced by c.

$$\begin{array}{cc} \begin{array}{ccccc} & & & k & \end{array} & \\ \begin{pmatrix} 1 & & & & \\ & \ddots & & c & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix} & \begin{array}{c} \\ j \\ \\ \\ \\ \end{array} \end{array}$$

Mathematically, for any **vector** $x = (x_1, \dots, x_n)'$, we denote

$$E_{jk}(c)x = (x_1, \dots, x_{j-1}, x_j + cx_k, x_{j+1}, \dots, x_n)'$$

We apply $E_{jk}(c)$ on both sides of the system $Ax = b$. Then the j-th equation

$$a'_{j\cdot}x = b_j$$

is replaced by

$$a'_{j\cdot}x + ca'_{k\cdot}x = b_j + cb_k.$$

The value of c depends on j and k, for the first column shown below, $c_j = -a_{j1}/a_{11}$.

1.2 Mathematical representations for GE

First, zeroing the first column

$$\begin{aligned} E_{21}(c_2^{(1)})Ax &= E_{21}(c_2^{(1)})b \\ E_{31}(c_3^{(1)})E_{21}(c_2^{(1)})Ax &= E_{31}(c_3^{(1)})E_{21}(c_2^{(1)})b \\ &\vdots \\ E_{n1}(c_n^{(1)})\cdots E_{31}(c_3^{(1)})E_{21}(c_2^{(1)})Ax &= E_{n1}(c_n^{(1)})\cdots E_{31}(c_3^{(1)})E_{21}(c_2^{(1)})b, \end{aligned}$$

where $c_j^{(1)} = -a_{j1}/a_{11}$. Denote

$$M_1 = E_{n1}(c_n^{(1)})\cdots E_{31}(c_3^{(1)})E_{21}(c_2^{(1)}).$$

Note that for $j > k$, $E_{jk}(c) = I + ce_je_k'$ is unit lower triangular and full rank and $E_{jk}^{-1}(c) = E_{jk}(-c)$, hence M_1 is a lower triangular matrix (homework).

We then apply the similar strategy to zero the k-th column for $k = 2, \dots, n-1$ sequentially. Finally, we obtain $Ux = \tilde{b}$, where

$$\begin{aligned} U &= M_{n-1}\dots M_1A, \\ \tilde{b} &= M_{n-1}\dots M_1b, \end{aligned}$$

Given a system of linear algebraic equations

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Step 1: Each row times a_{11}/a_{k1},
then use row one to subtract other rows.

$$\Rightarrow \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & \tilde{a}_{22} & \cdots & \tilde{a}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \tilde{a}_{n2} & \cdots & \tilde{a}_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \tilde{b}_2 \\ \vdots \\ \tilde{b}_n \end{bmatrix}$$

Step 2: The second row and down multiply by $\tilde{a}_{22}/\tilde{a}_{k2}$,
then use row two to subtract every row below.

$$\Rightarrow \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & \tilde{a}_{22} & \tilde{a}_{23} & \cdots & \tilde{a}_{2n} \\ 0 & 0 & \tilde{a}_{33} & \cdots & \tilde{a}_{3n} \\ \vdots & \vdots & \cdots & \ddots & \vdots \\ 0 & 0 & \tilde{a}_{n3} & \cdots & \tilde{a}_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \tilde{b}_2 \\ \tilde{b}_3 \\ \vdots \\ \tilde{b}_n \end{bmatrix}$$

Step 3: Similar to the previous two steps, repeat until all
elements in the lower triangle of the matrix A
become zeros.

$$\Rightarrow \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & \tilde{a}_{22} & \cdots & \tilde{a}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \tilde{a}_{n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \tilde{b}_2 \\ \vdots \\ \tilde{b}_n \end{bmatrix}$$

and each M_k has the shape

$$M_k = E_{n,k}(c_n^{(k)}) \dots E_{k+1,k}(c_{k+1}^{(k)}) = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & c_{k+1}^{(k)} & 1 & & \\ & & \vdots & & \ddots & \\ & & c_n^{(k)} & & & 1 \end{pmatrix},$$

where $c_i^{(k)} = -\tilde{a}_{ik}^{(k-1)}/\tilde{a}_{kk}^{(k-1)}$. Note that U an upper triangular matrix and M_k is unit lower triangular and full rank. The matrices M_ks are called the *Gauss transformations.*

1.3 LU decomposition

Let $L = M_1^{-1} \dots M_{n-1}^{-1}$, we have the decomposition

$$A = LU,$$

where M_k is lower triangular, so does M_k^{-1} and thus L is a lower triangular matrix and U is an upper triangular matrix.

Furthermore, by the Sherman-Morrison formula (homework) if

$$M_k = I + (0, \dots, 0, c_{k+1}^{(k)}, \dots, c_n^{(k)})' e_k',$$

then

$$M_k^{-1} = I - (0, \dots, 0, c_{k+1}^{(k)}, \dots, c_n^{(k)})' e_k'.$$

So the entries of L are simply $l_{jk} = -c_j^{(k)}$, $j > k$.

1.4 Flop counts for solving the linear system $Ax = b$

Now we can calculate the flop counts for solving the linear system $Ax = b$ ($A \in \mathbb{R}^{n \times n}$):

Step 1: The LU decomposition

- Each multiplier is computed with one division, the total cost is $\sum_{i=1}^{n} i = (n-1)n/2$
- At stage k, we need to modify a $(n-i) \times (n-i)$ matrix, each entry is modified by one subtraction and one multiplication, the total cost of the row operations is $2\sum_{i=1}^{n-1} i^2 = 2n^3/3 - n^2 + n/3$
- Total cost for LU is $\approx 2n^3/3$

Step 2: Given LU, forward substitution and backward substitution costs $2n^2$ flops

So, the total flops for solving linear equation $Ax = b$ costs $2n^3/3 + O(n^2)$ flops.

A few comments:

- LU decomposition exists if the principal sub-matrix $A[1:k, 1:k]$ is non-singular for $k = 1, \dots, n-1$.
- If the LU decomposition exists and A is non-singular, then the LU decomposition is unique and $det(A) = \prod_{i=1}^{n} u_{ii}$.
- For non-square matrix (rectangular matrix) $A \in \mathbb{R}^{m \times n}$. LU decomposition exists if $A[1:k, 1:k]$ is nonsingular for $k = \min\{m, n\}$. Then we can write it as

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 3 & 1 \\ 5 & 2 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ 0 & -2 \end{pmatrix}$$

and

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 4 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \end{pmatrix}$$

Then, one can slightly modify the original algorithm.

- In numpy: scipy.linalg.lu_factor() and scipy.linalg.lu_solve() is equivalent to numpy's numpy.linalg.solve()
- In R, LU is used in solve() function

In conclusion, avoid computing matrix inverse unless: 1) it is absolutely necessary to compute (e.g., obtain inverse of the covariance matrix); 2) n is small.

2 Pivoting for LU

Recall what we learned in the last lecture, what if we encounter a pivot $\tilde{a}_{kk}^{(k-1)}$ being 0 or close to 0 due to underflow? In this case, $c_i^{(k)} = -\tilde{a}_{ik}^{(k-1)}/\tilde{a}_{kk}^{(k-1)}$, which implies $c_i^{(k)} \approx \infty$.

Let's consider the example:

$$\begin{aligned} 0.0001x_1 + x_2 &= 1, \\ x_1 + x_2 &= 2, \end{aligned}$$

The solution is $x_1 = 1.0001$ and $x_2 = 0.9999$. Suppose we have 3 digits of precision, after the first step of elimination, we have

$$\begin{aligned} 0.0001x_1 + x_2 &= 1, \\ -10,000x_2 &= -10,000. \end{aligned}$$

Solving the linear system, one gets $x_2 = 1.000$ and $x_1 = 0.000$. This example shows that zero or very small pivots can cause trouble in computation.

To address this issue, we use pivoting. Let's recall last time, we denote $c_i^{(k)}$ as the multiplier. The idea behind pivoting is to make each $c_i^{(k)} < 1$: At the k-th stage the equation moves $\max_{i=k}^n |\tilde{a}_{ik}^{(k-1)}|$ to the k-th row.

We need to introduce *interchange permutations*. They are permutations obtained by sweeping two rows in the identity. Let's introducing matrix P, where

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

For a matrix $A \in \mathbb{R}^{3\times 3}$,

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

Note that PA is A with rows 1 and 3 interchanged and AP is A with columns 1 and 4 swapped; that is,

$$PA = \begin{pmatrix} a_{21} & a_{22} & a_{23} \\ a_{11} & a_{12} & a_{13} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}, \quad AP = \begin{pmatrix} a_{12} & a_{11} & a_{13} \\ a_{22} & a_{12} & a_{23} \\ a_{32} & a_{13} & a_{33} \end{pmatrix}.$$

Here is an example: Consider the matrix

$$A = \begin{pmatrix} 3 & 17 & 10 \\ 2 & 4 & -2 \\ 6 & 18 & -12 \end{pmatrix}$$

To get the smallest possible multipliers in the first GE, we need a_{11} to be the largest entry in the first column. So we introducing the permutation matrix

$$P_1 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Then,

$$P_1 A = \begin{pmatrix} 6 & 18 & -12 \\ 2 & 4 & -2 \\ 3 & 17 & 10 \end{pmatrix}$$

By calculation, we obtain

$$M_1 = \begin{pmatrix} 1 & 0 & 0 \\ -1/3 & 1 & 0 \\ -1/2 & 0 & 1 \end{pmatrix}$$

Then,

$$M_1 P_1 A = \begin{pmatrix} 6 & 18 & -12 \\ 0 & -2 & 2 \\ 0 & 8 & 16 \end{pmatrix}$$

Next, introducing

$$P_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

and

$$M_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1/4 & 1 \end{pmatrix}$$

Then,

$$M_2 P_2 M_1 P_1 A = \begin{pmatrix} 6 & 18 & -12 \\ 0 & 8 & 16 \\ 0 & 0 & 6 \end{pmatrix}$$

2.1 Partial pivoting for general $n \times n$ matrices

In general, here is how partial pivoting works:

for $k = 1 : n - 1$

Find an interchange permutation P_k that swaps $A[k,k]$ with the largest element in $|A[k:n,k]|$

Set $A = P_k A$

Determine the Gaussian transformation: M_k

if v is the k-th column of $M_k A$ **then**

Let $v[k+1:n] = 0$

end if

output: $U = M_{n-1} P_{n-1} \cdots M_1 P_1 A$

Again, U is an upper triangular matrix.

With partial pivoting, one can be show that $PA = LU$, where $P = P_{n-1} \cdots P_1$, L is a lower triangular with $|\ell_{ij}| \leq 1$. To solve $Ax = b$, we solve $PAx = Pb$ instead. Then, for the two triangular systems, we solve $Ly = Pb$ and $Ux = y$, This costs n^2 flops.

GE with partial pivoting is one of the most commonly used methods for solving general linear systems.

- In Python, linalg.solve() is a wrapper for the LAPACK routines dgesv (real-valued matrix) and zgesv (complex-valued matrix). It uses the LU decomposition with partial pivoting and row interchanges.
- In R, solve() uses LU decomposition. The Matrix package contains lu() function, which uses partial (row) pivoting

In addition to partial pivoting, there are other pivoting methods, e.g., complete pivoting (using both row and column pivoting) and rook pivoting (Poole and Neal, 2000).

2.2 Programming examples

numpy example:

```python
import pprint
import numpy as np
import scipy.linalg # Scipy linear algebra library

A = np.array([ [7, 3, -1, 2], [3, 8, 1, -4],
    [-1, 1, 4, -1], [2, -4, -1, 6] ])
P, L, U = scipy.linalg.lu(A)
pprint.pprint(P)
```

R example:

```r
install.packages("Matrix")
library(Matrix)
A <- matrix(c(7, 3, -1, 2, 3, 8, 1, -4,
   -1, 1, 4, -1, 2, -4, -1, 6), 4, 4)
luA <- lu(A)
elu <- expand(luA)
L <- elu$L
U <- elu$U
P <- elu$P
```

Question to consider: what is the difference between solve$(A)\% * \%b$ and solve(A, b) in R?

References

Poole, G. and L. Neal (2000). The Rook's pivoting strategy. *Journal of Computational and Applied Mathematics 123*, 353–369.