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

# Week 4-1: Pivoting and Cholesky decomposition

*Lecturer: Bo Y.-C. Ning* *April 19, 2022*



## Last time

- Gaussian elimination

## Today

- Pivoting
- Cholesky decomposition

## 1 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,

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

## 1.1 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 (**?**).

## 1.2 Programming examples

numpy example:

```
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)

A.same = P@L@U
```

R example:

```
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

A.same <- P%*%L%*%U
```

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

# 2 Cholesky decomposition

In the standard linear regression model, one might interest in solving the equation

$$X'X\beta = X'y,$$

where $X'X$ is symmetric and positive (semi-)definite.

We now know that LU is expansive. Thus whenever solving a problem, the structure should be exploited. Common structures include: symmetry, definiteness, sparsity, Kronecker product, band matrix, and low rank. For symmetric p.s.d. matrix, we can use Cholesky decomposition.

**Theorem 6.1** *Let $A \in \mathbb{R}^{n\times n}$ be symmetric and positive definite (p.d.). Then $A = LL'$ where $L$ is lower triangular with positive diagonal entries and is unique.*

Proof: (using induction).

If $n = 1$, then $L = \sqrt{a}$.

For $n > 1$, the block equation

$$\begin{pmatrix} a_{11} & a' \\ a & A_{22} \end{pmatrix} = \begin{pmatrix} \ell_{11} & 0'_{n-1} \\ \ell & L_{22} \end{pmatrix} \times \begin{pmatrix} \ell_{11} & \ell' \\ 0_{n-1} & L'_{22} \end{pmatrix}$$

The equation has a solution:

$$\ell_{11} = \sqrt{a_{11}}, \quad \ell = a/\ell_{11},$$
$$L_{22}L'_{22} = A_{22} - \ell\ell' = A_{22} - \ell\ell'.$$

Now $a_{11} > 0$, so $\ell_{11}$ and $\ell$ are uniquely determined. Also, $A_{22} - \ell\ell'$ is p.d. because $A$ is p.d. (why? check determinant). Hence, by induction hypothesis, $L_{22}$ exists and is unique.

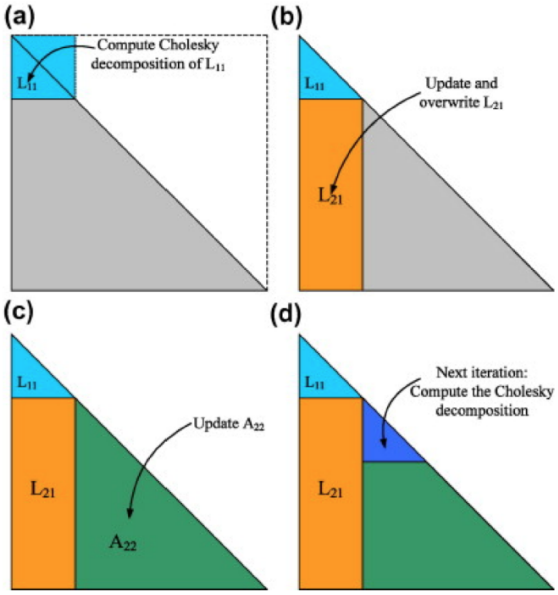
The proof above completely specifies the algorithm. The total flops for Cholesky decomposition is $[(n-1)^2 + (n-2)^2 + \cdots + 1^2]/2 \approx n^3/3 + O(n^2)$, which is approximately half the cost of LU decomposition.

In general, Cholesky decomposition is very stable. Failure of the decomposition simply means the matrix is not p.d.. Hence, this is a efficient way to test whether the matrix is p.d. or not. When zero pivots $a_{ii} = 0$ are encountered, we can still continue the algorithm by setting $\ell_{ii} = 0$ and $\ell = 0$, although a better way is using symmetric pivoting.

## 2.1 Cholesky with symmetric pivoting

We now consider the case when $A = LL'$ is positive semidefinite, but $\to L'$ is does not have full column rank.

$$\begin{pmatrix} a_{11} & a' \\ a & A_{22} \end{pmatrix} = \begin{pmatrix} \ell_{11} & 0'_{n-1} \\ \ell & L_{22} \end{pmatrix} \times \begin{pmatrix} \ell_{11} & \ell' \\ 0_{n-1} & L'_{22} \end{pmatrix}$$




with $\ell_{11} = \sqrt{a_{11}}, \ell = \frac{a}{\sqrt{a_{11}}}, L_{22}L_{22}' = A_{22} - \ell\ell'$.

In case of confusion, we denote $\tilde{a}_{kk}$ be the (1,1) element of $\tilde{A}_{kk} = L_{kk}L_{kk}'$. Note that $a_{11} = \tilde{a}_{11}$.

Symmetric pivoting: At each stage $k$, we permute both row and column such that $\max_{i=k}^{n} \tilde{a}_{kk}$ becomes the pivot. If we encounter $\max_{i=k}^{n} \tilde{a}_{kk} = 0$, then $\tilde{A}[k:n, k:n] = 0$ $P$ is the permutation matrix such that $PAP' = LL'$, where where $L \in \mathbb{R}^{n\times r}$, $r = \text{rank}(A)$.

In Python, np.linalg.cholesky() fails when matrix $A$ is not p.d., as it is based on Gauss elimination without pivoting. For example, try this code: np.linalg.cholesky([[1, 0], [0, 0]]) The solution is to use scipy.linalg.ldl() (? sytrf in LAPACK) instead.

In R, chol() is a wrapper function for LAPACK routines dpotrf (p.d.) and dpstrf (p.s.d with pivoting) with

- Option pivot = FALSE calls dpotrf, which does $A = LL'$ and gives error message if $A$ is not p.d.

- Option pivot = TRUE calls dpstrf, which does symmetric pivoting $PAP' = LL'$ and returns rank and pivot

- Option tol passes the tolerance to LAPACK for deciding zero pivots: default is $n\times$ machine epsilon $\times \max(\text{diag}(A))$.