Today:

Announcements:

- HW2 due, HW3 out
- Examlet | next week
- MOSS / collab. policy
- IEF

Fixing nonexistence of LU

What does LU with permutations process look like?

$$M_3 P_3 M_2 P_2 M_1 P_1 A = U$$

$$A = \underbrace{(M_3 P_3 M_2 P_2 M_1 P_1)^{-1}}_{L} U$$

Demo: LU with Partial Pivoting (Part I)

What about the L in LU?

Sort out what LU with pivoting looks like. Have: $M_3 P_3 M_2 P_2 M_1 P_1 A = U$.

$L_3 := M_3$ | $L_2 := P_3 M_2 P_3^{-1}$ | $L_1 := P_3 P_2 M_1 P_2^{-1} P_3^{-1}$

$\underline{L_3 L_2 L_1}\ \underline{P_3 P_2 P_1}$

$M_3\ P_3 M_2 \cancel{P_3^{-1}}\ \cancel{P_3} P_2 M_1 \cancel{P_2^{-1}} \cancel{P_3^{-1}}\ \cancel{P_3} \cancel{P_2}\ P_1 A = U$

$\underbrace{P_3 P_2 P_1}_{P} A = \underbrace{(L_1^{-1} L_2^{-1} L_3^{-1})}_{L} \cdot U$

$M_3 P_3 M_2 P_2 M_1 P_1 A = U$

$PA = LU$

Demo: LU with Partial Pivoting (Part II)

Computational Cost

$[AB]_{ij} = \sum_k A_{ik} B_{kj}$

What is the computational cost of multiplying two $n \times n$ matrices?

$O(n^3)$

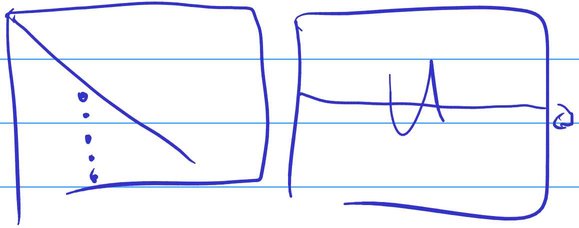
$M_1 A \to O(n^2)$

$\underbrace{PA}_{O(n)}$

What is the computational cost of carrying out LU factorization on an $n \times n$ matrix?

$\underbrace{O(n)}_{\#\text{ columns}} \cdot \underbrace{O(n^2)}_{\text{cost of applying an elim}} = O(n^3)$

Demo: Complexity of Mat-Mat multiplication and LU



More cost concerns

What's the cost of solving $A\mathbf{x} = \mathbf{b}$?

$O(n^3)$ for LU
$2 \cdot O(n^2)$ for fw/bw $\to O(n^2)$
$\Big\}\ O(n^3)$

What's the cost of solving $A\mathbf{x} = \mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$?

$O(n^3)$ for LU
$\to n \cdot O(n^2)$ for fw/bw
$\Big\}\ O(n^3)$

What's the cost of finding A^{-1}?

$AX = I$
$X = A^{-1}$

$PA = LU$
$A = P^T LU$
$LUx = Pb$

Solving w. m RHS:

$$O(n^3) + O(m \cdot n^2)$$

Cost: Worrying about the Constant, BLAS

$O(n^3)$ really means

$$\alpha \cdot n^3 + \beta \cdot n^2 + \gamma \cdot n + \delta.$$

All the non-leading and constants terms swept under the rug. But: at least the leading constant ultimately matters.

Getting that constant to be small is surprisingly hard, even for something deceptively simple such as matrix-matrix multiplication.

Idea: Rely on library implementation: *BLAS* (Fortran)

Level 1	$\mathbf{z} = \alpha\mathbf{x} + \mathbf{y}$	vector-vector operations $O(n)$ `?axpy`	s d c z
Level 2	$\mathbf{z} = A\mathbf{x} + \mathbf{y}$	matrix-vector operations $O(n^2)$ `?gemv`	
Level 3	$C = AB + \beta C$	matrix-matrix operations $O(n^3)$ `?gemm`, `?trsm`	

LAPACK

LAPACK: Implements 'higher-end' things (such as LU) using BLAS
Special matrix formats can also help save const significantly, e.g.

- banded
- sparse
- symmetric
- triangular

Sample routine names:

- `dgesvd`, `zgesdd`
- `dgetrf`, `dgetrs`

LU on Blocks: The Schur Complement

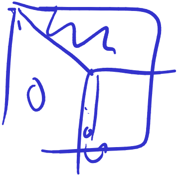
Given a matrix

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}, \quad -CA^{-1}$$

can we do 'block LU' to get a *block triangular matrix*?

$$\boxed{\begin{pmatrix} A & B \\ 0 & D-CA^{-1}B \end{pmatrix}}$$

LU: Special cases



A

inv.

What happens if we feed a non-invertible matrix to LU?

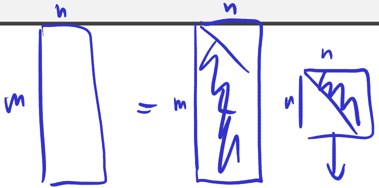
$PA = LU$ ← not inv.

inv. not invertible

What happens if we feed LU an $m \times n$ non-square matrices?

$A = LU$

$m \times n$ $m \times ?$ $? \times n$



$m > n$; $m \times n$ $n \times n$

$m < n$; $m \times m$ $m \times n$

Round-off Error in LU

Suppose $\epsilon_{mach} = 10^{-1}$
$\epsilon = 10^{-5}$

Consider factorization of $\begin{bmatrix} \epsilon & 1 \\ 1 & 1 \end{bmatrix}$ where $\epsilon < \epsilon_{\text{mach}}$:

- Without pivoting: $L = \begin{bmatrix} 1 & 0 \\ 1/\epsilon & 1 \end{bmatrix}$, $U = \begin{bmatrix} \epsilon & 1 \\ 0 & 1 - 1/\epsilon \end{bmatrix}$
  $1 + 10^5$
- Rounding: $\text{fl}(U)) = \begin{bmatrix} \epsilon & 1 \\ 0 & -1/\epsilon \end{bmatrix}$
- This leads to $L\,\text{fl}(U)) = \begin{bmatrix} \epsilon & 1 \\ 1 & 0 \end{bmatrix}$, a backward error of $\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$

Permuting the rows of A in partial pivoting gives $PA = \begin{bmatrix} 1 & 1 \\ \epsilon & 1 \end{bmatrix}$

- We now compute $L = \begin{bmatrix} 1 & 0 \\ \epsilon & 1 \end{bmatrix}$, $U = \begin{bmatrix} 1 & 1 \\ 0 & 1 - \epsilon \end{bmatrix}$, so $\text{fl}(U) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$
- This leads to $L\,\text{fl}(U) = \begin{bmatrix} 1 & 1 \\ \epsilon & 1 + \epsilon \end{bmatrix}$, a backward error of $\begin{bmatrix} 0 & 0 \\ 0 & \epsilon \end{bmatrix}$.

Changing matrices

$Ax=b$
$Ax=c$

$Ax=b$
$Cx=d$
$A+uv^T$

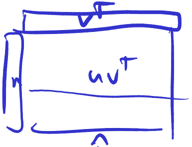
Seen: LU cheap to re-solve if RHS changes. (Able to keep the expensive bit, the LU factorization) What if the *matrix* changes?

$$(A+uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1+v^TA^{-1}u}$$

Demo: Sherman-Morrison

$A^{-1}\vec{x}$

solve $A\vec{z}=\vec{x}$



rank $uv^T=1$

In-Class Activity: LU

In-class activity: LU and Cost

Outline

What about non-square systems?

Specifically, what about linear systems with 'tall and skinny' matrices? (A: $m \times n$ with $m > n$) (aka *overdetermined* linear systems)

Specifically, any hope that we will solve those exactly?