Determining an LU factorization

Demo: LU Factorization [cleared]

Computational Cost

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

$O(n^3)$

- $u_{11} = a_{11}$, $\boldsymbol{u}_{12}^T = \boldsymbol{a}_{12}^T$.
- $\boldsymbol{l}_{21} = \boldsymbol{a}_{21}/u_{11}$.
- $L_{22}U_{22} = A_{22} - \boldsymbol{l}_{21}\boldsymbol{u}_{12}^T$.

O n flops

~~$O(n)$ strong n~~

$O(n^2)$

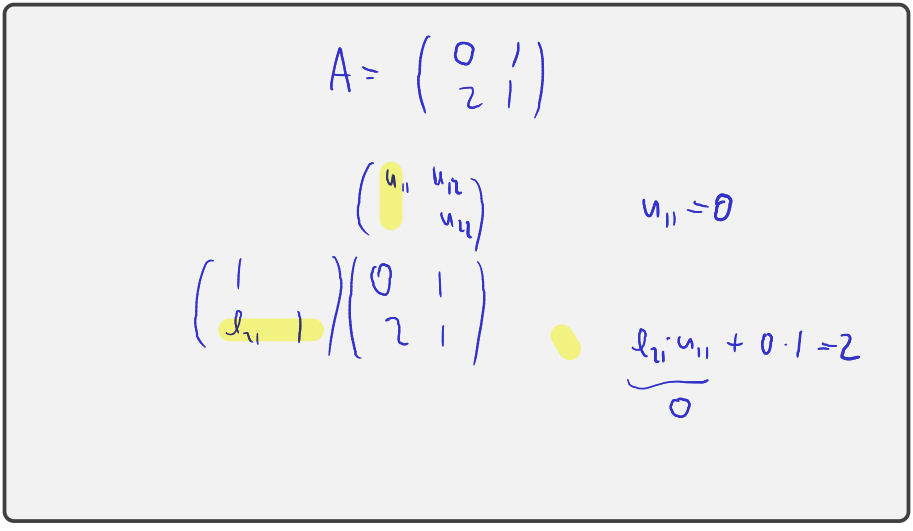
What is the computational cost of carrying out LU factorization on an $n \times n$ matrix?

$O(n^3)$

Demo: Complexity of Mat-Mat multiplication and LU [cleared]

LU: Failure Cases?

Is LU/Gaussian Elimination bulletproof?

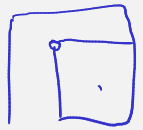


Saving the LU Factorization

What can be done to get something *like* an LU factorization?

Idea: control the magnitude of the denominator in LU, make as big as possible.

pivot



swap rows → partial pivoting

swap rows and cols → complete pivoting

Demo: LU Factorization with Partial Pivoting [cleared]

Cholesky: LU for Symmetric Positive Definite

LU *can* be used for SPD matrices. But can we do better?

$$A = L L^T$$

$$\begin{pmatrix} \ell_{11} & \\ \vec{\ell}_{21} & L_{22} \end{pmatrix} \begin{pmatrix} \ell_{11} & \vec{\ell}_{21}^T \\ & L_{22}^T \end{pmatrix} = \begin{pmatrix} A \\ \cdots \end{pmatrix}$$

$\ell_{11}^2 = a_{11} \geq 0 \iff A$ SPSD

↳ semi

- cheaper than LU
- no need for pivoting

More cost concerns

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

$LU\underbrace{x}_{y} = b$

$O(n^3)$ for LU, $O(n^2)$ for fw/bw

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

$O(n^3)$

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

$AX = I \hookrightarrow$

$Ax_1 = e_1$

$\vdots$

$Ax_n = e_n$

$Ax = b$

$x = A^{-1}b$

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.

Shrinking the constant: surprisingly hard (even for 'just' matmul)

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

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

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

Show (using perf): `numpy` matmul calls BLAS `dgemm`

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
- general

Sample routine names:

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

LU on Blocks: The Schur Complement

Given a matrix

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix},$$

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

Multiply top row $-CA^{-1}$, add to second

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

"Schur complement"

LU: Special cases

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

$PA = LU$ ✓ ✗ ✓ ✗

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

Round-off Error in LU without Pivoting

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