HW4

Exam 1 → starts Friday

Out of town 9/26, lecture as normal

Feedback

Computational Cost

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

$\boxed{O(n^3)}$

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

Cost$(n) = \alpha n^3 + \text{COT}$

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

$\boxed{O(n^3)}$

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

LU: Failure Cases?

Is LU/Gaussian Elimination bulletproof?

$$A = \begin{pmatrix} 0 & 1 \\ 2 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ \ell_{21} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 1 \end{pmatrix}$$

$u_{11} = 0$

$$\underbrace{\underset{0}{\cancel{u_{11}}} \cdot \ell_{21} + 1 \cdot 0}_{0} = 2$$

$\cancel{A = LU}$

$PA = LU$

permutation mat, permutes rows

all 0, 1

exactly one 1 per row/col

Saving the LU Factorization

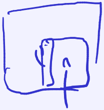
What can be done to get something *like* an LU factorization?

For num stability:

↳ swap so that u_{11} is as big as possible

the thing we divide by

$PA = LU$



← permute rows each step. typically sufficient

$PA = LU$ with row permutations: "partial" pivoting

$PAQ = LU$ row + col permutations: "complete" pivoting

↳ adds a leading order $O(n^3)$ cost term

Demo: LU Factorization with Partial Pivoting [cleared]

Saving the LU Factorization

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

Idea from linear algebra class: In Gaussian elimination, simply swap rows, equivalent linear system.

- Good idea: Swap rows if there's a zero in the way
- Even better idea: Find the largest entry (by absolute value), swap it to the top row.

The entry we divide by is called the *pivot*.

- Swapping rows to get a bigger pivot is called partial pivoting.
- Swapping rows *and columns* to get an even bigger pivot is called complete pivoting. (downside: additional $O(n^2)$ cost *per step* to find the pivot!)

Demo: LU Factorization with Partial Pivoting [cleared]

$PA = LU$

$SPA = SLU$

Cholesky: LU for Symmetric Positive Definite

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

$A = LL^T$

$$\begin{pmatrix} \ell_{11} & 0 \\ \vec{\ell}_{21} & L_{22} \end{pmatrix} \begin{pmatrix} \ell_{11} & \vec{\ell}_{21}^T \\ & L_{22}^T \end{pmatrix} = \begin{pmatrix} a_{11} & \vec{a}_{21}^T \\ \vec{a}_{21} & A_{22} \end{pmatrix}$$

$\ell_{11}^2 = a_{11} \rightsquigarrow \ell_{11} = \sqrt{a_{11}}$

$(\ell_{11} = 0 \Rightarrow A$ SP semi D$)$

$\ell_{11} \cdot \vec{\ell}_{21} = \vec{a}_{21} \to \vec{\ell}_{21} = \vec{a}_{21} / \ell_{11}$

$L_{22} L_{22}^T = A_{22} - \vec{\ell}_{21} \vec{\ell}_{21}^T$

$O(n^3)$

symm: $A = A^T$

PD: $\forall \vec{x} \in \mathbb{R}^n \setminus \{0\}: \vec{x}^T A \vec{x} > 0$

symm. matrices: eigenvalues real

PD: > 0

if $a_{11} < 0 \Rightarrow$ not SPD

$A = LL^T$

$x^T A x$

$= x^T L L^T x$

$= (L^T x)^T (L^T x)$

$= \|L^T \vec{x}\|_2^2$

$\|x\|_2^2 = 0$

diag nonz. $\Leftrightarrow$ L invertible: $\|L\vec{x}\|_2^2 \neq 0$

More cost concerns

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

① LU factor A $\to O(n^3)$
② Fw / bw subst $\to O(n^2)$
$\Big\}\ O(n^3)$

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

② $n \times$ fw / bw subst $\to O(n^3)$ $\Big\}\ O(n^3)$

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

$A \cdot A^{-1} = I$
$A \cdot X = I$ $\leftarrow$ solve col-by-col
$\Big\}\ O(n^3)$

Cost: Worrying about the Constant, BLAS

Basic Lin Alg Subroutine

$O(n^3)$ 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`

dgemm

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

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*?