3 LU factorization

Matrices can do neat things:

- Blur an image
- Traverse a graph
- Rotate geometry

Can we come up with a generic "undo" button for these things?

(... that does *not* depend on the application) (!)

To warm up, let's try this for matrices where it is super-easy.

Example: Upper triangular matrices

$$\begin{pmatrix} x & x & x & x \\ & x & x & x \\ & & x & x \\ & & & x \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$$

$a_{11} a + a_{12} b + a_{13} c + a_{14} d = x$ ④ solve for a

substitute

$a_{22} b + a_{23} c + a_{24} d = y$ ③ solve for b

substitute

$a_{33} c + a_{34} d = z$ ② solve for c

substitute

$a_{44} d = w$ ① solve for d

This is called "back-substitution".

Demo: Coding back-substitution

The analogous process for a lower triangular matrix is called "forward substitution".

What do we do about more general matrices?

In principle, same as in linear algebra class:

Gaussian elimination

Demo: Vanilla Gaussian Elimination

Leads to Row Echelon Form:

$$\begin{pmatrix} 2 & 4 & 1 & 5 & 7 & 10 & 11 \\ & 3 & 7 & 9 & 1 & 5 & 5 \\ & & & & 5 & 8 & 9 \\ & & & & & 7 & 4 \\ & & & & & & 3 \end{pmatrix}$$

Every row in REF is a linear combination of the original rows.

What's the difference between REF and an upper triangular matrix?

The REF matrix doesn't have to full down to the diagonal.

I.e. there are zeros allowed on and above the diagonal.

What happens if you don't just eliminate downward, but also upward?

What you get is called Gauss-Jordan elimination.

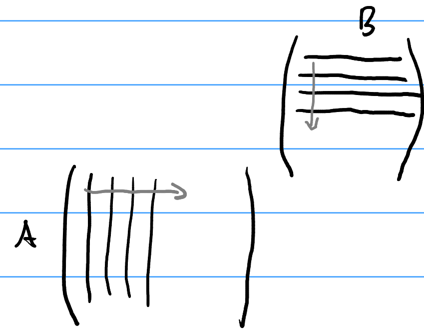
We won't look at it.

So we could implement Gaussian elimination every time we would like

w

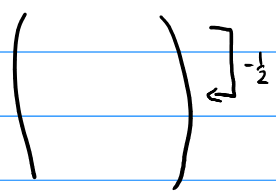
"store" the work we've put in, to reuse it with another right-hand side b?

Reconsider matrix-matrix multiplication.



Reading 1: Rows of B specify linear combinations of columns of A

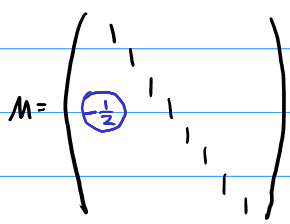
Reading 2: Columns of A specify linear combinations of rows of B

So: 

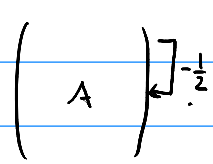
can be represented by matrix multiplication (from the left)

So how do you represent an elimination step as a matrix?

Idea: Start with an identity matrix...



...and add a single entry: add the first row *(-1/2) to the fourth.

So MA has the same result as 

Matrices like this are called elimination matrices.

Are elimination matrices invertible?

Example:

$$M = \begin{pmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ -\frac{1}{2} & & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{pmatrix}$$

Inverse has to be

$$M^{-1} = \begin{pmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ +\frac{1}{2} & & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{pmatrix}$$

to undo what M did.

Demo: Elimination matrices I

With enough elimination matrices, we should be able to arrive at REF...

$$M_\ell M_{\ell-1} \cdots M_2 M_1 A = U$$

"upper" row echelon form

What happens if we combine many elimination matrices like that?

Demo: Elimination matrices II

Summary:

- El.matrices with off-diagonal entries in a single column just "merge" when multiplied by one another.

- El.matrices with off-diagonal entries in different columns merge when we multiply (left-column) * (right-column) but not the other way around.

- Inverse: Flip sign below diagonal

We could rearrange that relationship to get a factorization of A!

$$A = \underbrace{M_1^{-1} M_2^{-1} \cdots M_{\ell-1}^{-1} M_\ell^{-1}}_{\text{Lower triangular matrix}} U.$$

$$\rightsquigarrow A = LU$$

This is called the LU factorization or LU decomposition.

Demo: LU factorization

Does an LU factorization help us solve Ax = b?

Certainly: Just plug the factorization in.

$$Ax=b \qquad A=LU$$

$$\Downarrow$$

$$\underbrace{LUx}_{\text{"y"}}=b$$

← a new unknown that we just invented

$$\Downarrow$$

$Ly=b$ ← solvable by forward substitution

→ now know y

$$\Downarrow$$

$Ux=y$ ← solvable by backward substitution

→ now know x

$$\Downarrow$$

$Ax=b$ solved.

So is LU/Gaussian elimination bulletproof?

No, it's actually quite fragile. Consider this example:

$$\begin{pmatrix} 0 & 1 \\ 2 & 1 \end{pmatrix} \quad \left(-\frac{2}{0}\right)$$

bad.

So is our process just too stupid to find the LU factorization?

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

$$\begin{pmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{pmatrix} \quad \rightsquigarrow \quad u_{11} = 0$$

$$\rightsquigarrow \quad \underbrace{u_{11} \cdot l_{21}}_{0} + 1 \cdot 0 = 2 \quad ↯$$

$$\begin{pmatrix} 1 & 0 \\ l_{21} & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 2 & 1 \end{pmatrix}$$

Nope, no LU factorization exists.

How are we going to fix this mess?

Idea: Find a nonzero entry, swap it into the top row.

Dividing by small numbers can produce very large numbers.
Floating point numbers do not work very well with numbers whose magnitude varies a lot. So we'll try to avoid that.
(We'll talk more about floating point later in the class.)

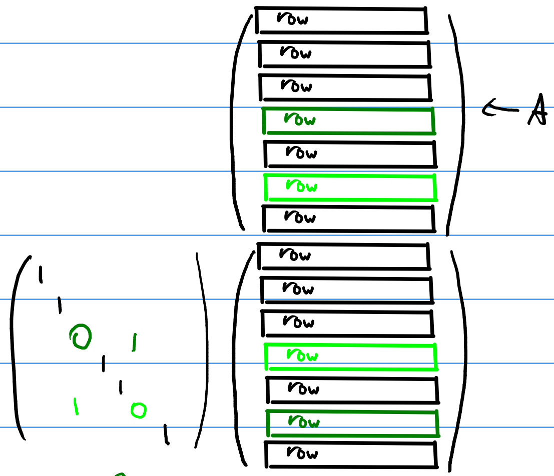
$\rightsquigarrow$ Even better idea: Find the largest entry (by absolute value), swap it into the top row.

This idea is called "partial pivoting" or "row pivoting".

The entry that ends up in the top left is called the "pivot".

How do we swap rows in matrix notation?

Using so-called permutation matrices.



$\rightsquigarrow PA$ is A with swapped rows

permutation matrix P

What's the inverse of a permutation matrix?

The permutation matrix itself. Makes sense if you think about it. (swap->swap)

How do we combine partial pivoting with the elimination matrices?

1	$P_1 A$	Pivot first column
2	$M_1 P_1 A$	Eliminate first column
3	$P_2 M_1 P_1 A$	Pivot second column
4	$M_2 P_2 M_1 P_1 A$	Eliminate second column
5	$P_3 M_2 P_2 M_1 P_1 A$	Pivot third column
6	$M_3 P_3 M_2 P_2 M_1 P_1 A$	Eliminate third column

That has made quite a mess of our LU factorization, right?

$$M_3 P_3 M_2 P_2 M_1 P_1 A = U \quad \Big| \; M_3^{-1}\cdot$$

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

$$\vdots$$

$$A = \underbrace{P_1 M_1^{-1} P_2 M_2^{-1} P_3 M_3^{-1}}_{\text{Is this still lower triangular?}} U$$

Demo: LU with Pivoting (Part I)

No, this is actually no longer lower triangular. Oops.

So... how do we sort out this mess?

Best hope: Try to get to a factorization of the form $PA = LU$

where P is a product of permutation matrices.

Unfortunately, we can't just move all the P's to the left and

all the M's to the right, past one another. (They don't "commute.")

So... how do we sort out this mess? (cont'd)

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

Define: $L_3 = M_3$

$L_2 = P_3 M_2 P_3^{-1}$

$L_3 = P_3 P_2 M_1 P_2^{-1} P_3^{-1}$

Then: $L_3 L_2 L_1 P_3 P_2 P_1$

$= (M_3)(P_3 M_2 \cancel{P_3^{-1}})(\cancel{P_3} P_2 M_1 \cancel{P_2^{-1}} \cancel{P_3^{-1}}) \cancel{P_3} \cancel{P_2} P_1$

$= M_3 P_3 M_2 P_2 M_1 P_1$!

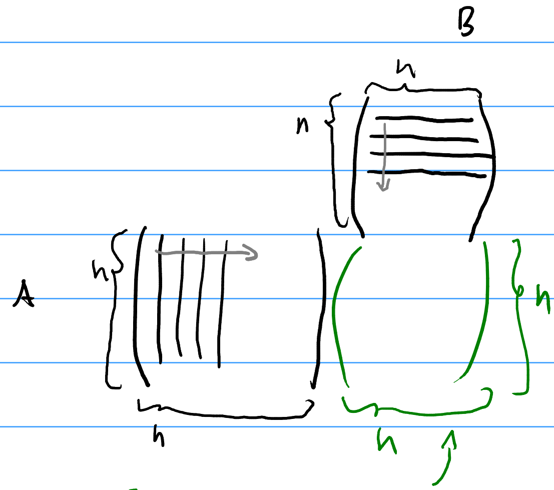
And perhaps the best miracle of them all is that L_1, L_2, L_3 are still lower triangular!

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

$PA = LU$

Demo: LU with Partial Pivoting (Part II)

Let's talk about computational cost. What is the asymptotic cost of multiplying two n x n matrices?




```
for i in range(n):
  for j in range(n):
    for k in range(n):
      AB[i,j] += A[i,k]*B[k,j]
```

three nested for-loops (of length n)

n^2 entries, each takes O(n) work to compute $\longrightarrow O(n^3)$

So how expensive is LU factorization?

$$\underbrace{M_3P_3M_2P_2M_1P_1A}_{2n\cdot n^3 \sim n^4\ ?!!} = U$$

$\longleftarrow$ Fortunately not.

Multiplications with permutation matrices and elimination matrices can be carried out in $O(n^2)$. (Why?)

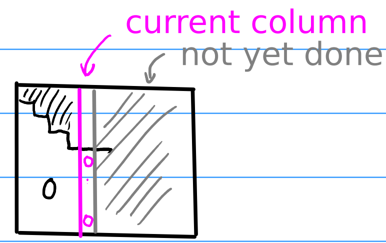
So the overall cost of LU is "just" $O(n^3)$.

Demo: Complexity of Mat-Mat multiplication and LU

Are there any remaining failure scenarios for LU?

The largest below-diagonal entry is zero (or close to it).

-> We don't have a valid pivot.



Not a problem: Column is already "upper triangular"!

Just move to next column. (But keep current row.)

-> Pivoted LU cannot fail.

End result: U upper triangular

What happens if the matrix in LU factorization is not invertible?

invertible (a product of elimination matrices)

not invertible (otherwise: contradiction!)

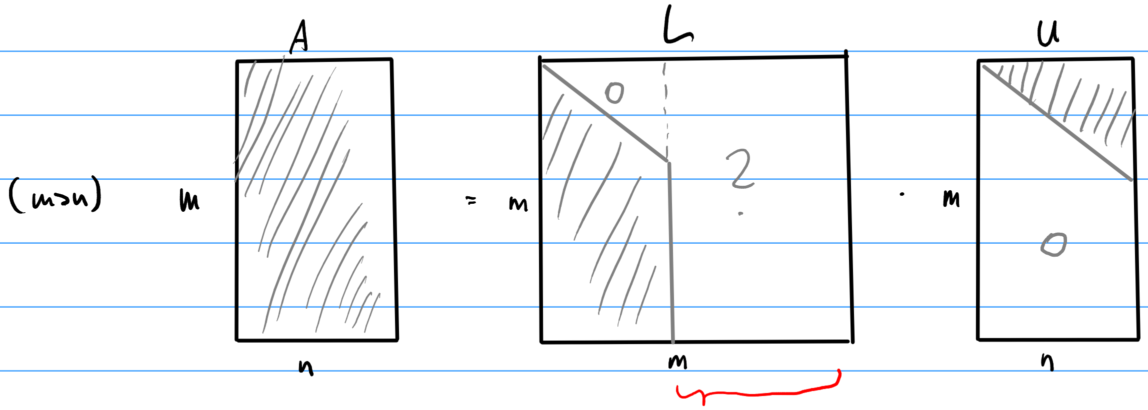
$$PA = LU$$

not invertible (by assumption)

invertible (by definition of permutation matrix)

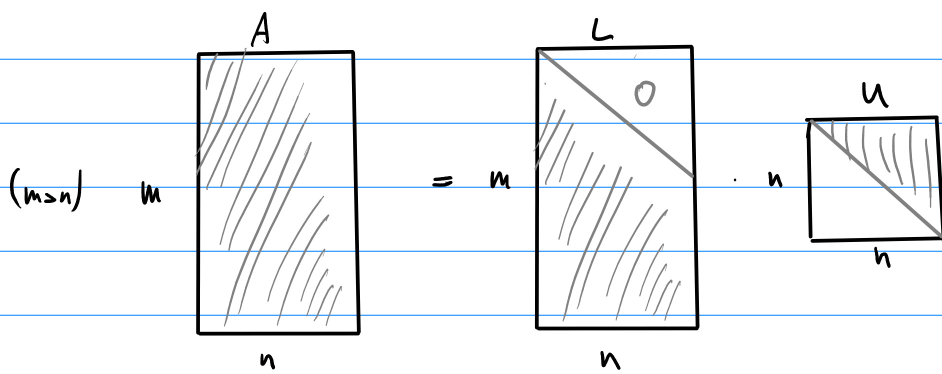
Can LU deal with non-square matrices?

Sure! There are four possibilities.

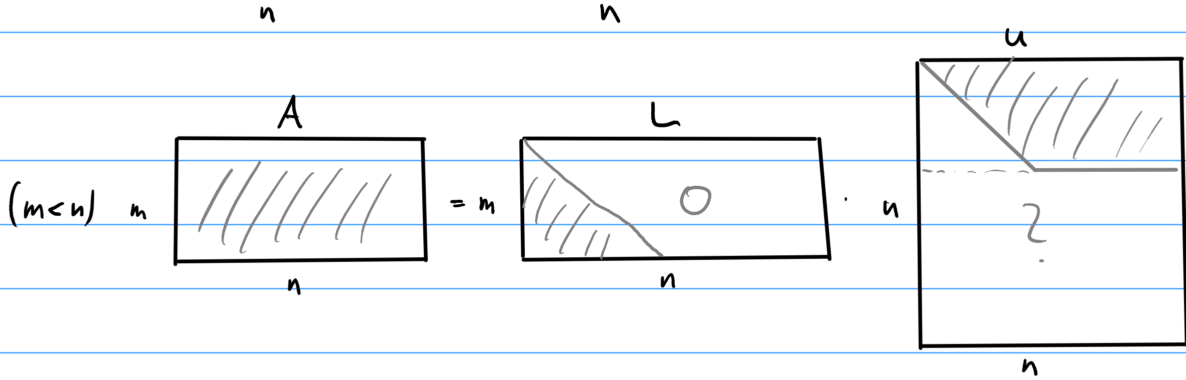


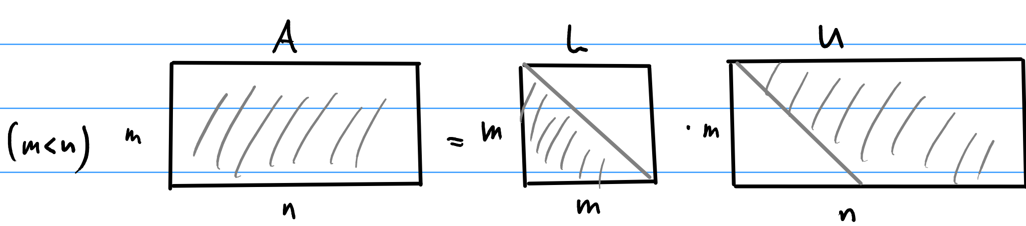
("full" LU)

does not matter:
multiplied by 0s in U



("reduced" LU)





Software will typically produce the "reduced" version.

(which is clearly more efficient!)