Computational Linear Algebra: Ax=b (Part II: LU and PA=LU Decompositions)

David Shuman

February 15, 2022

LU Decomposition

- Two steps to solving an $n \times n$ equation $Ax = b$:
  1. Gaussian elimination: expensive: $O(n^3)$
  2. Back substitution: less expensive: $O(n^2)$

LU Decomposition

- Two steps to solving an $n \times n$ equation $Ax = b$:
  1. Gaussian elimination: expensive: $O(n^3)$
  2. Back substitution: less expensive: $O(n^2)$

- Sometimes in practice you need to solve

$$Ax = b_1, \quad Ax = b_2, \quad Ax = b_3, \quad \dots, \quad Ax = b_M$$

where the A is the same each time and M is large (engineering, dynamical systems)

LU Decomposition

- Two steps to solving an $n \times n$ equation $Ax = b$:
  1. Gaussian elimination: expensive: $O(n^3)$
  2. Back substitution: less expensive: $O(n^2)$

- Sometimes in practice you need to solve

$$Ax = b_1, \quad Ax = b_2, \quad Ax = b_3, \quad \dots, \quad Ax = b_M$$

  where the A is the same each time and M is large (engineering, dynamical systems)

- The same row reductions are done on A each time, only the augmented part b changes

- It would be dumb to do this problem over and over again

- Idea: "store" the elimination steps and apply them to b

LU Decomposition

We will take our $n \times n$ matrix and *decompose* or *factorize* it as the product

$$A = \underbrace{\begin{bmatrix} 1 & & & & & \\ * & 1 & & & & \\ * & * & 1 & & & \\ * & * & * & 1 & & \\ * & * & * & * & 1 & \\ * & * & * & * & * & 1 \end{bmatrix}}_{Lower\ \ unit-triangular} \underbrace{\begin{bmatrix} * & * & * & * & * & * \\ & * & * & * & * & * \\ & & * & * & * & * \\ & & & * & * & * \\ & & & & * & * \\ & & & & & * \end{bmatrix}}_{Upper\ \ triangular}$$

Eg., (we will show *how* this is done soon)

$$\begin{pmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 4 & 3 & 1 & 0 \\ 3 & 4 & 1 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

Solving $Ax = b$ with LU Decomposition

- Factorize $A = LU$
- Then solve $Ax = LUx = b$ in two steps:
  1. Let $y = Ux$, and solve $Ly = b$
  2. Solve $y = Ux$ to find x

How do we perform the LU decomposition?

- Examples and general technique on board

Time Complexity of the LU Decomposition

- Putting A into its LU factorization takes one application of Gaussian elimination $\approx \frac{2}{3}n^3$ operations for an $n \times n$ matrix
- Solving $LUx = b$ requires 2 back substitutions — one to solve $Ly = b$ for y and one to solve $Ux = y$ for x. This takes $2n^2$ operations
- To solve $Ax = b_1, Ax = b_2, Ax = b_3, \ldots, Ax = b_M$ takes
  - On the order of $\frac{2}{3}n^3$ to perform $A = LU$
  - $2Mn^2$ to perform the $2M$ back substitutions
  - On the order of $\frac{2}{3}n^3 + 2Mn^2$ total
- To naively use Gaussian elimination on M problems of the form $Ax = b$ requires on the order of
  - $M * \frac{2}{3}n^3$ flops
  - If M is close to n this is like n^4
- R's solve (and the corresponding Matlab backslash function) uses an optimized version of LU to solve $Ax = b$

Existence of an LU Decomposition

- Does an LU decomposition exist for all matrices?

Existence of an LU Decomposition

- Does an LU decomposition exist for all matrices?
  - No, consider

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ a & 1 \end{pmatrix} \begin{pmatrix} b & c \\ 0 & d \end{pmatrix} = \begin{pmatrix} b & c \\ ab & ac + d \end{pmatrix}$$

Existence of an LU Decomposition

- Does an LU decomposition exist for all matrices?
  - No, consider

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ a & 1 \end{pmatrix} \begin{pmatrix} b & c \\ 0 & d \end{pmatrix} = \begin{pmatrix} b & c \\ ab & ac + d \end{pmatrix}$$

- Which step in our algorithm is problematic?

Existence of an LU Decomposition

- Does an LU decomposition exist for all matrices?
  - No, consider

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ a & 1 \end{pmatrix} \begin{pmatrix} b & c \\ 0 & d \end{pmatrix} = \begin{pmatrix} b & c \\ ab & ac + d \end{pmatrix}$$

- Which step in our algorithm is problematic?
  - When we reach a zero pivot
  - We'll come back to this problem later to look at a workaround

Existence of an LU Decomposition

- Does an LU decomposition exist for all matrices?
  - No, consider

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ a & 1 \end{pmatrix} \begin{pmatrix} b & c \\ 0 & d \end{pmatrix} = \begin{pmatrix} b & c \\ ab & ac + d \end{pmatrix}$$

- Which step in our algorithm is problematic?
  - When we reach a zero pivot
  - We'll come back to this problem later to look at a workaround
- When does a square matrix A have an LU decomposition?

Existence of an LU Decomposition

- Does an LU decomposition exist for all matrices?
  - No, consider

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ a & 1 \end{pmatrix} \begin{pmatrix} b & c \\ 0 & d \end{pmatrix} = \begin{pmatrix} b & c \\ ab & ac + d \end{pmatrix}$$

- Which step in our algorithm is problematic?
  - When we reach a zero pivot
  - We'll come back to this problem later to look at a workaround

- When does a square matrix A have an LU decomposition?
  - If and only if the upper-left sub-blocks $A_{1:k,1:k}$ are non-singular for all $1 \leq k \leq n$

Uniqueness of LU Decomposition

Is the LU decomposition for a matrix A unique?

- i.e., can a matrix have 2 (or more) LU decompositions?

Uniqueness of LU Decomposition

Is the LU decomposition for a matrix A unique?

- i.e., can a matrix have 2 (or more) LU decompositions?

Yes it is unique (no it cannot have multiple LU decompositions)

Assume there exist two pairs of lower and upper triangular matrices such that

$$A = L_1 U_1 = L_2 U_2$$

Then

$$L_2^{-1} L_1 = U_2 U_1^{-1} = I$$

because the left-hand side is lower triangular and the right-hand side is upper-triangular (each is closed under inversion and product). So $L_1 = L_2$ and $U_1 = U_2$.

Activity: Implementation of LU Decomposition

Pseudocode:

$$U = A, L = I$$

$$\text{for } k = 1 \text{ to } n-1$$

$$\quad \text{for } j = k+1 \text{ to } n$$

$$\qquad \ell_{jk} = \frac{u_{jk}}{u_{kk}}$$

$$\qquad u_{j,k:n} = u_{j,k:n} - \ell_{jk} u_{k,k:n}$$

Two Sources of Error with "Naive" Gaussian Elimination

- Sometimes we have a 0 in the pivot position: $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

Two Sources of Error with "Naive" Gaussian Elimination

- Sometimes we have a 0 in the pivot position: $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
- Swamping

Two Sources of Error with "Naive" Gaussian Elimination

- Sometimes we have a 0 in the pivot position: $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
- Swamping
  - Let ε be less than ε_M; try this with $\varepsilon = 10^{-20}$

$$\begin{pmatrix} \varepsilon & 1 \\ 1 & 1 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 \\ 1/\varepsilon & 1 \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} \varepsilon & 1 \\ 0 & 1 - 1/\varepsilon \end{pmatrix}}_{U}$$

$$\longrightarrow \quad \underbrace{\begin{pmatrix} 1 & 0 \\ 1/\varepsilon & 1 \end{pmatrix}}_{\tilde{L}} \underbrace{\begin{pmatrix} \varepsilon & 1 \\ 0 & -1/\varepsilon \end{pmatrix}}_{\tilde{U}} = \begin{pmatrix} \varepsilon & 1 \\ 1 & 0 \end{pmatrix} \text{ in floating point arithmetic}$$

Two Sources of Error with "Naive" Gaussian Elimination

- Sometimes we have a 0 in the pivot position: $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
- Swamping
  - Let ε be less than ε_M; try this with $\varepsilon = 10^{-20}$

$$\begin{pmatrix} \varepsilon & 1 \\ 1 & 1 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 \\ 1/\varepsilon & 1 \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} \varepsilon & 1 \\ 0 & 1 - 1/\varepsilon \end{pmatrix}}_{U}$$

$$\longrightarrow \quad \underbrace{\begin{pmatrix} 1 & 0 \\ 1/\varepsilon & 1 \end{pmatrix}}_{\tilde{L}} \underbrace{\begin{pmatrix} \varepsilon & 1 \\ 0 & -1/\varepsilon \end{pmatrix}}_{\tilde{U}} = \begin{pmatrix} \varepsilon & 1 \\ 1 & 0 \end{pmatrix} \text{ in floating point arithmetic}$$

  - Problem goes away if we swap rows

$$\begin{pmatrix} 1 & 1 \\ \varepsilon & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \varepsilon & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 - \varepsilon \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \varepsilon & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ \varepsilon & 1 \end{pmatrix}$$

Two Sources of Error with "Naive" Gaussian Elimination

- Sometimes we have a 0 in the pivot position: $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
- Swamping
  - Let ε be less than ε_M; try this with $\varepsilon = 10^{-20}$

$$\begin{pmatrix} \varepsilon & 1 \\ 1 & 1 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 \\ 1/\varepsilon & 1 \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} \varepsilon & 1 \\ 0 & 1 - 1/\varepsilon \end{pmatrix}}_{U}$$

$$\longrightarrow \quad \underbrace{\begin{pmatrix} 1 & 0 \\ 1/\varepsilon & 1 \end{pmatrix}}_{\tilde{L}} \underbrace{\begin{pmatrix} \varepsilon & 1 \\ 0 & -1/\varepsilon \end{pmatrix}}_{\tilde{U}} = \begin{pmatrix} \varepsilon & 1 \\ 1 & 0 \end{pmatrix} \text{ in floating point arithmetic}$$

  - Problem goes away if we swap rows

$$\begin{pmatrix} 1 & 1 \\ \varepsilon & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \varepsilon & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 - \varepsilon \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \varepsilon & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ \varepsilon & 1 \end{pmatrix}$$

  - Moral: multipliers should be kept small in Gaussian elimination

Pivoting: Complete and Partial

- No particular reason why A_{kk} must be chosen as the k^{th} pivot

Pivoting: Complete and Partial

- No particular reason why A_{kk} must be chosen as the k^{th} pivot
- We want to pick the largest magnitude element amongst candidates

Pivoting: Complete and Partial

- No particular reason why A_{kk} must be chosen as the k^{th} pivot
- We want to pick the largest magnitude element amongst candidates
- Can do so by exchanging rows (pivoting)

Pivoting: Complete and Partial

- No particular reason why A_{kk} must be chosen as the k^{th} pivot
- We want to pick the largest magnitude element amongst candidates
- Can do so by exchanging rows (pivoting)
- Why not search through all remaining rows and columns for best pivot (complete pivoting)?

Pivoting: Complete and Partial

- No particular reason why A_{kk} must be chosen as the k^{th} pivot
- We want to pick the largest magnitude element amongst candidates
- Can do so by exchanging rows (pivoting)
- Why not search through all remaining rows and columns for best pivot (complete pivoting)?
  - Would incur complexity cost of $O(n^3)$ just to find the next pivot
  - Too expensive

Pivoting: Complete and Partial

- No particular reason why A_{kk} must be chosen as the k^{th} pivot
- We want to pick the largest magnitude element amongst candidates
- Can do so by exchanging rows (pivoting)
- Why not search through all remaining rows and columns for best pivot (complete pivoting)?
  - Would incur complexity cost of $O(n^3)$ just to find the next pivot
  - Too expensive
- Standard method: partial pivoting
  - Choose from the $n - k + 1$ subdiagonal elements in column k
  - Overall complexity cost of $O(n^2)$, which is acceptable

$PA = LU$ Example

Important points:

- Permutation matrices P_i have a single 1 in each row and each column
- Exchange rows by multiplying on the left by a permutation matrix
- Choose the k^{th} pivot to be the largest magnitude element from the $n - k + 1$ subdiagonal elements in column k
- Swap the pivot row with row k
- Leads to $L_{n-1}P_{n-1}L_{n-2}P_{n-2}\dots L_2P_2L_1P_1A = U$
- Luckily, $L_{k+1}P_{k+1}L_kP_k = L'_{k+1}L'_kP_{k+1}P_k$
- Product of permutation matrices is another permutation matrix
- We have already seen that taking the inverse of the product of a sequence of these special L_k matrices yields a unit lower-triangular matrix
- Therefore, we arrive at $PA = LU$

$PA = LU$ (cont.)

Important points (cont.):

- All entries in the final L are less than 1 in magnitude
- Complexity with partial pivoting is the same as without pivoting: $O(n^3)$
- Any square matrix can be factorized this way

$PA = LU$ (cont.)

Important points (cont.):

- All entries in the final L are less than 1 in magnitude
- Complexity with partial pivoting is the same as without pivoting: $O(n^3)$
- Any square matrix can be factorized this way

Solving $Ax = b$:

- $PAx = LUx = Pb$
- Solve $Ly = Pb$ for y
- Solve $Ux = y$ for x
- Expensive part is still the elimination step, and so a single factorization can be used to efficiently solve for multiple choices of b

Pseudocode of $PA = LU$ Decomposition

Pseudocode:

$U = A, L = I, P = I$

for $k = 1$ to $n - 1$

  Select $i \geq k$ to maximize $|U_{ik}|$

  $U_{k,k:n} \leftrightarrow U_{i,k:n}$ (interchange two rows)

  $\ell_{k,1:k-1} \leftrightarrow \ell_{i,1:k-1}$

  $P_{k,:} \leftrightarrow P_{i,:}$

  for $j = k + 1$ to n (contents of this for loop are the same as before)

$$\ell_{jk} = \frac{U_{jk}}{U_{kk}}$$

$$U_{j,k:n} = U_{j,k:n} - \ell_{jk} U_{k,k:n}$$

Stability of Gaussian Elimination with Pivoting?

- We saw an example in naive Gaussian elimination where the fact that $||L||||U|| >> ||A||$ caused numerical errors
- With pivoting, all entries in L are small, so $||L||$ is always small
- Define *growth factor*: $\rho := \frac{\max_{i,j} |U_{ij}|}{\max_{i,j} |A_{ij}|}$
- If ρ is small, $||U||$ is on the same order as $||A||$, and algorithm is (backward) stable
- Example of worst-case instability ($\rho = 2^{n-1}$):

$$\begin{pmatrix} 1 & & & & 1 \\ -1 & 1 & & & 1 \\ -1 & -1 & 1 & & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & & & & \\ -1 & 1 & & & \\ -1 & -1 & 1 & & \\ -1 & -1 & -1 & 1 & \\ -1 & -1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & & & & 1 \\ & 1 & & & 2 \\ & & 1 & & 4 \\ & & & 1 & 8 \\ & & & & 16 \end{pmatrix}$$

- For practical problems, this never happens and Gaussian elimination with pivoting ($PA = LU$) is extremely stable