COS 302 Precept 5

Spring 2020

Princeton University

Outline

LU Decomposition

QR Decomposition

Outline

LU Decomposition

QR Decomposition

LU Decomposition

Definition

LU decomposition is a procedure for decomposing an $n \times n$ matrix A into a product of a lower-triangular matrix L and an upper triangular matrix U:

$$LU = A$$

Example

Let's write out $LU = A$ explicitly for a 3×3 matrix. Then we have elementwise:

$$\begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}.$$

LU Decomposition

Here's a particular choice of l_{ij}, u_{kl} that satisfies the definition:

$$\begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & -1 & 1 & \\ & & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & & \\ & 1 & -1 & \\ & & 1 & -1 \\ & & & 1 \end{bmatrix} = \begin{bmatrix} 1 & -1 & & \\ -1 & 2 & -1 & \\ & -1 & 2 & -1 \\ & & -1 & 2 \end{bmatrix}$$

Why would we do this?

- Notice that when $LU = A$, we can write out the solution of linear system of equations in a convenient form: $Ax = (LU)x = L(Ux) = b$. We can split the solve into two parts, both involving only triangular matrices.
- Why would we do this?
- Triangular matrices are very fast to solve!

Interpretation as a Linear System (1 of 2)

Setting $\mathbf{Ux} = \mathbf{y}$, we get the expression $\mathbf{Ly} = \mathbf{b}$:

$$\begin{bmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

Notice we can easily solve for $\mathbf{y}$ with the equivalent linear system:

$$\begin{array}{ccccccccc} l_{11}y_1 & & & & & & & = b_1 \\ l_{21}y_1 & + & l_{22}y_2 & & & & & = b_2 \\ \vdots & & \vdots & & \ddots & & & \vdots \\ l_{n1}y_1 & + & l_{n2}y_2 & + & \cdots & + & l_{nn}y_n & = b_n \end{array}$$

Interpretation as a Linear System (2 of 2)

We then solve $\mathbf{Ux} = \mathbf{y}$, using the $\mathbf{y}$ we just found:

$$\begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{21} & \dots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Notice we can easily solve for $\mathbf{y}$ with the equivalent linear system:

$$\begin{array}{ccccccccc} u_{11}x_1 & + & u_{12}x_2 & + & \cdots & + & u_{nn}x_n & = & y_1 \\ & & & & \ddots & \vdots & & & \vdots \\ & & & & u_{n-1,n}x_{n-1} & + & u_{nn}x_n & = & y_{n-1} \\ & & & & & & u_{nn}x_n & = & y_n \end{array}$$

General Algorithm

We formalize this procedure through the following recipe:

Overview

First set $\mathbf{Ux} = \mathbf{y}$, and solve an intermediate expression for $\mathbf{y}$: $\mathbf{Ly} = \mathbf{b}$. This is called *forward substitution*. Then, given the solution $\mathbf{y}$, we solve $\mathbf{Ux} = \mathbf{y}$. This is called *back substitution*.

General Algorithm

Forward substitution

Forward substitution is a recursive recipe: we use the results of each previous step to reduce computation in the following one:

$$
\begin{aligned}
y_1 &= \frac{b_1}{l_{11}} \\
y_i &= \frac{1}{l_{ii}} \left(b_i - \sum_{j=1}^{i-1} l_{ij} y_j \right)
\end{aligned}
$$

for each $i = 2, \ldots, N$.

General Algorithm

Back substitution

Given the result of the previous step **fy**, we now solve $\mathbf{Ux} = \mathbf{y}$ for $\mathbf{x}$:

$$x_N = \frac{y_N}{u_{NN}}$$
$$x_i = \frac{1}{u_{ii}} \left(y_i - \sum_{j=i+1}^{N} u_{ij} x_j \right)$$

for each $i = N - 1, \ldots, 1$.

Outline

LU Decomposition

QR Decomposition

QR Decomposition

Definition

QR decomposition of an $n \times n$ square matrix A decomposes it into the product of an orthogonal matrix Q (i.e., $Q^T Q = I$) and an upper triangular matrix R:

$$A = QR$$

This factorization is unique when A is invertible and R has positive elements along the diagonal.

Why would we do this?

- Like LU decomposition, QR decomposition allows easier solving of linear systems.
- We've already encountered the convenience of upper triangular matrices in linear systems (simple back substitution).
- In addition, orthogonal matrices are trivial to invert (the transpose equals the inverse).

Solving linear systems

Let $A \in \mathbb{R}^{n \times n}$ be invertible with factorization QR.

$$\begin{aligned} Ax &= b \\ QRx &= b \\ (Q^T Q)Rx &= Q^T b \\ Rx &= Q^T b \end{aligned}$$

Setting $y = Q^T b$, we can now use back substitution to solve $Rx = y$.

QR for overdetermined systems

Recall that when we have more equations than unknowns, we cannot solve our system exactly. Instead, we can use "least squares" to minimize the sum of squared errors.

Let $A \in \mathbb{R}^{m \times n}$ be the coefficient matrix for an overdetermined linear system, i.e., $m > n$. We want to minimize

$$\|Ax - b\|^2 = \sum_{i=1}^{m} (A_i x - b)^2$$

QR for overdetermined systems

When A has linearly independent columns, the inverse of $A^T A$ exists, and we can find the minimizing solution with the normal equation:

$$\begin{aligned} A^T A x &= A^T b \\ x &= (A^T A)^{-1} A^T b \end{aligned}$$

QR for overdetermined systems

Definition

For a rectangular matrix $A \in \mathbb{R}^{m \times n}$, QR decomposition decomposes A into an orthogonal matrix $Q \in \mathbb{R}^{m \times m}$ and an upper triangular matrix $R \in \mathbb{R}^{m \times n}$ where the bottom $m - n$ rows contain zeroes.

QR for overdetermined systems

We can rewrite the normal equation solution using our QR decomposition:

$$
\begin{aligned}
x = \left(A^T A\right)^{-1} A^T b &= \left((QR)^T (QR)\right)^{-1} (QR)^T b \\
&= \left(R^T Q^T Q R\right)^{-1} R^T Q^T b \\
&= \left(R^T R\right)^{-1} R^T Q^T b \\
&= R^{-1} R^{-T} R^T Q^T b \\
&= R^{-1} Q^T b
\end{aligned}
$$

QR for overdetermined systems

This makes solving certain ill-conditioned systems more numerically stable. For example, when A's columns are almost linearly dependent, numerical problems introduced when inverting $A^T A$ are avoided with QR.

Example

Assume we round scalars to 8 significant decimal digits.

$$A = \begin{bmatrix} 1 & -1 \\ 0 & 10^{-5} \\ 0 & 0 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 10^{-5} \\ 1 \end{bmatrix}$$

$$A^T A = \begin{bmatrix} 1 & -1 \\ -1 & 1 + 10^{-10} \end{bmatrix} \rightsquigarrow \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

$A^T A$ is singular after rounding, making inversion impossible.

Example from `http://www.seas.ucla.edu/~vandenbe/133A/`

Example

Instead, with QR decomposition:

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad R = \begin{bmatrix} 1 & -1 \\ 0 & 10^{-5} \end{bmatrix}$$

Now $R^{-1}Q^Tb$ can be computed without introducing any rounding errors.

Example from `http://www.seas.ucla.edu/~vandenbe/133A/`