LU Factorization

Gaussian Elimination and Matrix Factorization

Consider the process of Gaussian elimination applied to the $n \times n$ matrix

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}.$$

At the beginning of the k^{th} stage of Gaussian elimination, we have

$$A^{(k)} = \begin{bmatrix} a_{11}^{(1)} & & \cdots & & a_{1n}^{(1)} \\ & \ddots & & & \vdots \\ & & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ & & \vdots & \ddots & \vdots \\ & & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)} \end{bmatrix}$$

The goal at this stage is to introduce zeros in column k beneath the diagonal. So we will be modifying entries in rows $i = k+1, \ldots, n$ and columns $j = k, \ldots, n$, by means of row operations

$$(\text{row } i) \leftarrow (\text{row } i) - m_{ik}\ (\text{row } k)$$

where m_{ik} is chosen strategically to eliminate the entries in column k, beneath row k. That is, we are setting

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} * a_{kj}^{(k)} \tag{1}$$

for each row i where

$$m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \tag{2}$$

so that $a_{ik}^{(k)}$ is eliminated ($a_{ik}^{(k+1)} = a_{ik}^{(k)} - m_{ik} * a_{kk}^{(k)} = 0$).

We can represent these row operations as the following matrix multiplication:

$$\underbrace{\begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & -m_{k+1,k} & 1 & & \\ & & \vdots & & \ddots & \\ & & -m_{nk} & & & 1 \end{bmatrix}}_{M^{(k)}} \underbrace{\begin{bmatrix} a_{11}^{(1)} & & \cdots & & a_{1n}^{(1)} \\ & \ddots & & & \vdots \\ & & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ & & a_{k+1,k}^{(k)} & \cdots & a_{k+1,n}^{(k)} \\ & & \vdots & \ddots & \vdots \\ & & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)} \end{bmatrix}}_{A^{(k)}} = \underbrace{\begin{bmatrix} a_{11}^{(1)} & & & & \cdots & a_{1n}^{(1)} \\ & \ddots & & & & \vdots \\ & & a_{kk}^{(k)} & & \cdots & a_{kn}^{(k)} \\ & & 0 & a_{k+1,k+1}^{(k+1)} & \cdots & a_{k+1,n}^{(k+1)} \\ & & \vdots & \vdots & \ddots & \vdots \\ & & 0 & a_{n,k+1}^{(k+1)} & \cdots & a_{nn}^{(k+1)} \end{bmatrix}}_{A^{(k+1)}}$$

The zeros are shown explicitly to make it clear where we have eliminated entries. The matrix $M^{(k)}$ performs the k^{th} stage of Gaussian elimination on $A^{(k)}$ and the result is $A^{(k+1)}$. Note that $M^{(k)}$ is the result of performing the row operations on the identity matrix.

What is the inverse of $M^{(k)}$? The matrix $M^{(k)}$ says "take m_{ik} times row k and and subtract it from row i". Then the inverse of this operation is simply to "add it back" (i.e. remove the negative signs).

$$\left(M^{(k)}\right)^{-1} = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & m_{k+1,k} & 1 & & \\ & & \vdots & & \ddots & \\ & & m_{nk} & & & 1 \end{bmatrix}$$

We can write

$$A^{(k)} = \left(M^{(k)}\right)^{-1} A^{(k+1)}$$

So $M^{(k)}$ performs the row operations on $A^{(k)}$, while $\left(M^{(k)}\right)^{-1}$ "factors" the row operations out of $A^{(k)}$.

After all of the Gaussian elimination steps, we have

$$M^{(n-1)} \cdots M^{(2)} M^{(1)} A = A^{(n)} = U$$

where U is the resulting upper triangular matrix (the n^{th} stage of Gaussian elimination is trivial since there are no rows beneath the last row.) Equivalently, we can write

$$A = \left(M^{(1)}\right)^{-1} \left(M^{(2)}\right)^{-1} \cdots \left(M^{(n-1)}\right)^{-1} U = LU$$

where

$$L = \left(M^{(1)}\right)^{-1} \left(M^{(2)}\right)^{-1} \cdots \left(M^{(n-1)}\right)^{-1}.$$

By accumulating the matrix operations according to this sequence we get

$$L = \begin{bmatrix} 1 & & & & \\ m_{21} & 1 & & & \\ \vdots & & \ddots & & \\ & & & 1 & \\ m_{n1} & \cdots & & m_{n,n-1} & 1 \end{bmatrix}$$

and we have the factorization $A = LU$ where L is lower triangular and U is upper triangular.

Cost of LU Factorization

What is the cost of the LU factorization in terms of *flops* (floating point operations)? Consider the number of multipliers to be computed (according to (2)) at each stage of Gaussian elimination:

At the first stage there are $n-1$ multipliers to compute.
At the second stage there are $n-2$ multipliers to compute.
$\vdots$
At the last stage there is 1 multiplier to compute.

Each multiplier is computed with one division. Therefore the cost of computing all of the multipliers is[1]

$$\sum_{k=1}^{n-1} k = \frac{1}{2}(n-1)n.$$

Now, consider the cost of the row operations. At each stage k, we are modifying a $(n-k)\times(n-k)$ block of entries according to (1). (Note that I have not included the eliminated entries, since they will be zero by design and do not need to be computed.) The block size at each stage decreases from $(n–1)\times(n–1)$ at the first stage, to 1×1 at the last. Each entry is modified by one subtraction and one multiplication. Therefore the cost of the row operations, assuming the multipliers have already been computed, is[1]

$$\begin{aligned} 2\sum_{k=1}^{n-1} k^2 &= 2\frac{1}{6}(n-1)n(2(n-1)+1) \\ &= \frac{2}{3}n^3 - n^2 + \frac{1}{3}n. \end{aligned}$$

Adding this to the flop count for the multiplier computations, we see that the total cost of the LU factor work is

$$\frac{2}{3}n^3 + O(n^2).$$

---

[1]

$$\sum_{i=1}^{n} i = \frac{1}{2}n(n+1)$$

$$\sum_{i=1}^{n} i^2 = \frac{1}{6}n(n+1)(2n+1)$$

Using LU factorization to solve linear systems

The LU factorization is very useful for linear system solving. Once the factorization is obtained, it can be used to solve a system $Ax = b$ repeatedly with different right-hand side vectors, without having to repeat the process of Gaussian elimination each time. The system $Ax = b$ is solved in two stages by doing the following.

1. Factorize $A = LU$
2. Solve $Ly = b$ for y, using *forward substitution*.
3. Solve $Ux = y$ for x, using *backward substitution*.

This method works because

$$\begin{aligned} Ax &= b \\ LUx &= b \\ L(Ux) &= b \\ Ly &= b. \end{aligned}$$

To solve a new system of the form $Ax = c$, only steps 2 and 3 would need to be repeated. Steps 2 and 3 are refered to as the triangular solves.

<u>Forward Substitution</u> ($Ly = b$)

$$\begin{bmatrix} 1 & & & & \\ \ell_{21} & 1 & & & \\ \vdots & & \ddots & & \\ \ell_{n-1,1} & & \cdots & 1 & \\ \ell_{n1} & & \cdots & \ell_{n,n-1} & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}$$

$$\begin{aligned} y_1 &= b_1 \\ y_2 &= b_2 - \ell_{21} y_1 \\ &\vdots \\ y_n &= b_n - \ell_{n1} y_1 - \cdots - \ell_{n,n-1} y_{n-1} \end{aligned}$$

What is the cost of forward substitution, in terms of flops (floating point operations)? There are $i - 1$ subtractions and $i - 1$ multiplications at each step $i = 1, \ldots, n$. Therefore the flop count is

$$\begin{aligned} 2 \sum_{i=1}^{n} (i-1) &= 2 \left(\sum_{i=1}^{n} i - \sum_{i=1}^{n} 1 \right) \\ &= 2 \left(\frac{1}{2} n(n+1) - n \right) \\ &= n^2 - n. \end{aligned}$$

Backward Substitution ($Ux = y$)

$$\begin{bmatrix} u_{11} & u_{12} & & \cdots & u_{1n} \\ & u_{22} & & \cdots & u_{2n} \\ & & \ddots & & \vdots \\ & & & u_{n-1,n-1} & u_{n-1,n} \\ & & & & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix}$$

$$\begin{aligned} x_n &= y_n / u_{nn} \\ x_{n-1} &= (y_{n-1} - u_{n-1,n} x_n) / u_{n-1,n-1} \\ &\vdots \\ x_1 &= (y_1 - u_{12} x_2 - \cdots - u_{1n} x_n) / u_{11} \end{aligned}$$

There is one division at each step, and there are $i-1$ subtractions and $i-1$ multiplications at each step $i = 1, \ldots, n$. Therefore the flop count is

$$\begin{aligned} n + 2 \sum_{i=1}^{n} (i-1) &= n + 2 \left(\sum_{i=1}^{n} i - \sum_{i=1}^{n} 1 \right) \\ &= n + 2 \left(\frac{1}{2} n(n+1) - n \right) \\ &= n^2. \end{aligned}$$

Row Pivoting

What if at some stage k we have $a_{kk}^{(k)} = 0$? Then we cannot form the multiplier $m_{ik} = a_{ik}^{(k)}/a_{kk}^{(k)}$ for any i. In fact, even if $a_{kk}^{(k)}$ is nonzero but close to zero, we can run into numerical issues. If $a_{kk}^{(k)} \approx 0$ then the multiplier m_{ik} is very large, and this can lead to problems such as (i) the magnification of round-off error in entries when multiplying them by m_{ik}, or (ii) "swamping", where large values overpower or "swamp" small values when being added to or subtracted from them, leading to incorrect results. The multipliers in Guassian elimination should be kept as small as possible to avoid these kinds of problems. This can be accomplished by swapping rows.

The following protocol is known as row pivoting, also called partial pivoting (as opposed to *complete* pivoting, where rows *and* columns are swapped). At each stage k, choose row ℓ such that

$$|a_{\ell k}^{(k)}| = \max_{i=k,\dots,n} |a_{ik}^{(k)}|.$$

Swap this row with row k, and continue with the elimination. This ensures that the multipliers will be no greater than 1 in absolute value.

Note: If $a_{kk}^{(k)} = 0$, and no nonzero entries can be found directly below it, that means the rows are not all linearly independent, and the matrix is singular (not invertible). It may still be possible to obtain an LU factorization of the matrix in this case, although it will be useless. There will be some zeros on the diagonal of U and it will not be possible to use the factorization to solve a system $Ax = b$ (see next section), which is the primary purpose of the LU factorization.

Row permutations can be represented as matrix operations using permutation matrices. A permutation matix is created by applying row permutations to the identity matix. If P is a permutation matrix (P is the same as I except the rows are permuted), then PA is the same as A, except the rows are permuted in the same way. So the row permutations are performed on A by multiplying it by P.

The Gaussian elimination process with row pivoting takes the form

$$M^{(n-1)}P^{(n-1)} \cdots M^{(2)}P^{(2)}M^{(1)}P^{(1)}A = U$$

Note that in cases where the diagonal entry is already the largest one ($\ell = k$) the permutation matrix is just the identity ($P^{(k)} = I$). The inverse of a single row exchange is the same row exchange, so these permutation matrices are each their own inverse. We can therefore write

$$A = P^{(1)} \left(M^{(1)}\right)^{-1} P^{(2)} \left(M^{(2)}\right)^{-1} \cdots P^{(n-1)} \left(M^{(n-1)}\right)^{-1} U.$$

At this point it's not very clear how to separate the permutations from the eliminations. However, by a stroke of good luck, it turns out that these operations on A can be reordered in a convenient way. Let

$$P = P^{(n-1)} \cdots P^{(2)}P^{(1)}.$$

So P represents the accumulation of row pivot permutations, in the order that they occur. We have

$$\begin{aligned} PA &= P^{(n-1)} \cdots P^{(2)}P^{(1)}\ P^{(1)} \left(M^{(1)}\right)^{-1} P^{(2)} \left(M^{(2)}\right)^{-1} \cdots P^{(n-1)} \left(M^{(n-1)}\right)^{-1} U \\ &= P^{(n-1)} \cdots P^{(2)} \left(M^{(1)}\right)^{-1} P^{(2)} \left(M^{(2)}\right)^{-1} \cdots P^{(n-1)} \left(M^{(n-1)}\right)^{-1} U \end{aligned}$$

Now for a clever trick. I'm going to insert $P^{(3)} \cdots P^{(n-1)} P^{(n-1)} \cdots P^{(3)} = I$ beside $\left(M^{(2)}\right)^{-1}$:

$$
\begin{aligned}
PA &= P^{(n-1)} \cdots P^{(2)} \left(M^{(1)}\right)^{-1} P^{(2)} \left[P^{(3)} \cdots P^{(n-1)} P^{(n-1)} \cdots P^{(3)}\right] \left(M^{(2)}\right)^{-1} \cdots P^{(n-1)} \left(M^{(n-1)}\right)^{-1} U \\
&= \left[P^{(n-1)} \cdots P^{(2)} \left(M^{(1)}\right)^{-1} P^{(2)} P^{(3)} \cdots P^{(n-1)}\right] P^{(n-1)} \cdots P^{(3)} \left(M^{(2)}\right)^{-1} \cdots P^{(n-1)} \left(M^{(n-1)}\right)^{-1} U
\end{aligned}
$$

Repeating this trick gives a useful grouping of the matrices. It is much easier see what is going on in a simple example. If $n = 4$ we have

$$
PA = P^{(3)} P^{(2)} \left(M^{(1)}\right)^{-1} P^{(2)} \left(M^{(2)}\right)^{-1} P^{(3)} \left(M^{(3)}\right)^{-1} U
$$

Insert $P^{(3)} P^{(3)} = I$ beside the $\left(M^{(2)}\right)^{-1}$, and group the matrices:

$$
\begin{aligned}
PA &= P^{(3)} P^{(2)} \left(M^{(1)}\right)^{-1} P^{(2)} \left[P^{(3)} P^{(3)}\right] \left(M^{(2)}\right)^{-1} P^{(3)} \left(M^{(3)}\right)^{-1} U \\
&= \left[P^{(3)} P^{(2)} \left(M^{(1)}\right)^{-1} P^{(2)} P^{(3)}\right] \left[P^{(3)} \left(M^{(2)}\right)^{-1} P^{(3)}\right] \left(M^{(3)}\right)^{-1} U
\end{aligned}
$$

In general, we can express PA as a product of groups of the form:

$$
P^{(n-1)} \cdots P^{(k+1)} \left(M^{(k)}\right)^{-1} P^{(k+1)} \cdots P^{(n-1)}
$$

for $k = 1, \ldots, n-1$. i.e. each matrix $M^{(k)}$ is sandwiched between a sequence of permutations P_j with $j > k$. The resulting matrix of each group has the same structure as $M^{(k)}$ except the multipliers in column k are permuted according to the permutation sequence. For example:

$$
\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 \\ m_{21} & 1 & 0 \\ m_{31} & 0 & 1 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}
=
\begin{bmatrix} 1 & 0 & 0 \\ m_{31} & 0 & 1 \\ m_{21} & 1 & 0 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}
=
\begin{bmatrix} 1 & 0 & 0 \\ m_{31} & 1 & 0 \\ m_{21} & 0 & 1 \end{bmatrix}
$$

As you can see, the permutations applied on the left side permute the rows, and the permutations applied on the right side put the 1's back on the diagonal. We always end up with a lower triangular matrix like we had before, but with the multipliers swapped. Therefore we can write $PA = LU$ where L consists of the elimination factors with the multipliers swapped accordingly.

This result has a simple interpretation. Gaussian elimination with row pivoting is equivalent to the following procedure:

1. Permute the rows of A according to P.
2. Apply Gaussian elimination without pivoting to PA.

Of course, the process is not carried out this way in practice since P is not known ahead of time. To obtain the $PA = LU$ factorization, carry out Gaussian elimination with row pivoting, swapping the previous multipliers in the lower triangular factors as well as the rows in $A^{(k)}$. Accumulate the row permutations in a matrix P, and in the end you will have the factorization $PA = LU$.

Using PA=LU to solve linear systems

The $PA = LU$ factorization can be used to solve a linear system $Ax = b$ as follows.

1. Factorize: $PA = LU$
2. Solve $Ly = Pb$ for y, using forward substitution.
3. Solve $Ux = y$ for x, using backward substitution.

This method works because

$$\begin{aligned} Ax &= b \\ PAx &= Pb \\ LUx &= Pb \\ L(Ux) &= Pb \\ Ly &= Pb. \end{aligned}$$

Example

Use the $PA = LU$ factorization with row pivoting to solve the system $Ax = b$ where

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 1 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 6 \\ 15 \\ 16 \end{bmatrix}.$$

1. Factorize $PA = LU$

Swap rows 1 and 3

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

row 2 $\leftarrow$ row 2 $- \frac{4}{7}$(row 1)
row 3 $\leftarrow$ row 3 $- \frac{1}{7}$(row 1)

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{4}{7} & 1 & 0 \\ \frac{1}{7} & 0 & 1 \end{bmatrix} \begin{bmatrix} 7 & 8 & 1 \\ 0 & \frac{3}{7} & \frac{38}{7} \\ 0 & \frac{6}{7} & \frac{20}{7} \end{bmatrix}$$

Swap rows 2 and 3

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{7} & 1 & 0 \\ \frac{4}{7} & 0 & 1 \end{bmatrix} \begin{bmatrix} 7 & 8 & 1 \\ 0 & \frac{6}{7} & \frac{20}{7} \\ 0 & \frac{3}{7} & \frac{38}{7} \end{bmatrix}$$

row 3 $\leftarrow$ row 3 $- \frac{1}{2}$(row 2)

$$\underset{P}{\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}} \underset{A}{\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 1 \end{bmatrix}} = \underset{L}{\begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{7} & 1 & 0 \\ \frac{4}{7} & \frac{1}{2} & 1 \end{bmatrix}} \underset{U}{\begin{bmatrix} 7 & 8 & 1 \\ 0 & \frac{6}{7} & \frac{20}{7} \\ 0 & 0 & 4 \end{bmatrix}}$$

2. Forward substitution ($Ly = Pb$)

$$\begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{7} & 1 & 0 \\ \frac{4}{7} & \frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 6 \\ 15 \\ 16 \end{bmatrix} = \begin{bmatrix} 16 \\ 6 \\ 15 \end{bmatrix}$$

$$\begin{aligned} y_1 &= 16 \\ y_2 &= 6 - \frac{1}{7}16 = \frac{26}{7} \\ y_3 &= 15 - \frac{4}{7}16 - \frac{1}{2}\frac{26}{7} = 4 \end{aligned}$$

2. Backward substitution ($Ux = y$)

$$\begin{bmatrix} 7 & 8 & 1 \\ 0 & \frac{6}{7} & \frac{20}{7} \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 16 \\ \frac{26}{7} \\ 4 \end{bmatrix}$$

$$\begin{aligned} x_3 &= \frac{4}{4} = 1 \\ x_2 &= \left(\frac{26}{7} - \frac{20}{7} \cdot 1\right)\frac{7}{6} = 1 \\ x_1 &= (16 - 8 \cdot 1 - 1 \cdot 1)\,\frac{1}{7} = 1 \end{aligned}$$

Thus, the solution is

$$x = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$