Sage and Linear Algebra Worksheet

FCLA Section PDM

Robert Beezer
Department of Mathematics and Computer Science
University of Puget Sound

Spring 2019

1 LU Decomposition, Triangular Form

This is a topic not covered in our text. You *can* find a discussion in *A Second Course in Linear Algebra* at http://linear.ups.edu/scla/html/index.html.

Our goal is to row-reduce a matrix with elementary matrices, track the changes, and arrive at an expression for a square matrix A as a product of a lower-triangular matrix, L, and an upper-triangular matrix, U, that is

$$A = LU$$

the so-called **LU decomposition**. I sometimes prefer to call it **triangular form**.

There are no exercises in this worksheet, but instead there is a careful and detailed exposition of using elementary matrices (row operations) to arrive at a **matrix decomposition**. There are many kinds of matrix decompositions, such as the **singular value decomposition** (SVD). Five or six such decompositions form a central part of the linear algebra canon. Again, see *A Second Course in Linear Algebra* for details on these.

We decompose a 5×5 matrix. It is most natural to describe an LU decomposition of a square matrix, but the decomposition can be generalized to rectangular matrices.

```
entries =
A = matrix(QQ, [[-6, -10,  0, 10, 14],
                [ 2,   3,  0, -4, -3],
                [ 0,  -2, -3,  1,  8],
                [ 5,   6, -3, -7, -3],
                [-1,   1,  6, -1, -8]])
A
```

Elementary matrices to "do" row operations in the first column.

```
actionA = elementary_matrix(QQ, 5, row1=1, row2=0,
    scale=-2)*elementary_matrix(QQ, 5, row1=3, row2=0,
    scale=-5)*elementary_matrix(QQ, 5, row1=4, row2=0,
    scale=1)*elementary_matrix(QQ, 5, row1=0, scale=-1/6)
B = actionA*A
B
```

Now in second column, moving to **row-echelon form** (i.e. *not* **reduced row-echelon form**).

```
actionB = elementary_matrix(QQ, 5, row1=2, row2=1,
    scale=2)*elementary_matrix(QQ, 5, row1=3, row2=1,
    scale=7/3)*elementary_matrix(QQ, 5, row1=4, row2=1,
    scale=-8/3)*elementary_matrix(QQ, 5, row1=1, scale=-3)
C = actionB*B
C
```

The "bottom" of the third column.

```
actionC = elementary_matrix(QQ, 5, row1=3, row2=2,
    scale=3)*elementary_matrix(QQ, 5, row1=4, row2=2,
    scale=-6)*elementary_matrix(QQ, 5, row1=2, scale=-1/3)
D = actionC*C
D
```

And now the penultimate column.

```
actionD = elementary_matrix(QQ, 5, row1=4, row2=3,
    scale=-2)*elementary_matrix(QQ, 5, row1=3, scale=1)
E = actionD*D
E
```

And done.

```
actionE = elementary_matrix(QQ, 5, row1=4, scale=1)
F = actionE*E
F
```

Clearly, `F` has determinant 1, since it is an upper triangular matrix with diagonal entries equal to 1. By tracking the effect of the above manipulations (tantamount to performing row operations) we expect that

$$\det(A) = \left(\frac{1}{-1/6}\right)\left(\frac{1}{-3}\right)\left(\frac{1}{-1/3}\right)\left(\frac{1}{1}\right)\left(\frac{1}{1}\right)\det(F) = -6.$$

Let's check.

```
A.determinant()
```

Yep. But it gets better. `F` is the product of the "action" matrices on the left of `A`.

```
total_action = prod([actionE, actionD, actionC, actionB,
    actionA])
total_action
```

Notice that the elementary matrices we used are all lower triangular (because we just formed zeros below the diagonal of the original matrix as we brought it to row-echelon form, and there were no row swaps). Hence their product is again lower triangular. Now check that we have the correct matrix.

```
F == total_action * A
```

The "total action" matrix is a product of elementary matrices, which are individually nonsingular. So their product is nonsingular. Futhermore, the inverse is again lower triangular.

```
ta_inv = total_action.inverse()
ta_inv
```

We reach our goal by rearranging the equality above, writing `A` as a product of a lower-triangular matrix with an upper-triangular matrix.

```
A == ta_inv * F
```

Yes! So we have decomposed the original matrix (`A`) into the product of a lower triangular matrix (inverse of the total action matrix) and an upper triangular matrix with all ones on the diagonal (`F`, the original matrix in row-echelon form).

```
A, ta_inv, F
```

This decomposition (the **LU decomposition**) can be useful for solving systems quickly. You **forward solve** with L, then **back solve** with U.

More specifically, suppose you want to solve $A\mathbf{x} = \mathbf{b}$ for $\mathbf{x}$, and you have a decomposition $A = LU$. First solve the intermediate system, $L\mathbf{y} = \mathbf{b}$ for $\mathbf{y}$, which can be accomplished easily by determining the entries of $\mathbf{y}$ in order, exploiting the lower triangular nature of L. This is what is meant by the term **forward solve**.

With a solution for $\mathbf{y}$, form the system $U\mathbf{x} = \mathbf{y}$. You can check that a solution, $\mathbf{x}$, to this system is also a solution to the original system $A\mathbf{x} = \mathbf{b}$. Further, this solution can be found easily by determining the entries of $\mathbf{x}$ in reverse order, exploiting the upper triangular nature of U. This is what is meant by the term **back solve**.

We solve *two* simple systems, but only do half as many row-operations as if we went fully to reduced row-echelon form. If you count the opertions carefully, you will see that this is a big win, roughly reducing computation time by a factor of half for large systems.