LU and upper echelon form

In [1]:

```
#keep
import numpy as np
import scipy.linalg as la
```

Part I: The Problem

Here's a matrix:

In [2]:

```
#keep
A = np.array([[0.,  0., -1., -1.,  0.,  0.,  0., -1.,  0.],
              [0.,  0.,  0.,  0.,  0., -1.,  0.,  1.,  0.],
              [0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
              [-1.,  0.,  1.,  0.,  0.,  0., -1.,  0.,  0.],
              [1., -1.,  0.,  1., -1.,  1.,  0.,  0., -1.],
              [0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.],
              [0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.],
              [0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
              [0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.]])
```

In [3]:

```
#keep
P, L, U = la.lu(A)
```

Check that it is actually a factorization:

In [4]:

```
#keep
la.norm(A-P.dot(L).dot(U))
```

Out[4]:

```
0.0
```

Now look at `U`:

In [5]:

```
U
```

Out[5]:

```
array([[-1.,  0.,  1.,  0.,  0.,  0., -1.,  0.,  0.],
       [ 0., -1.,  1.,  1., -1.,  1., -1.,  0., -1.],
       [ 0.,  0., -1., -1.,  0.,  0.,  0., -1.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0., -1.,  1., -1., -1., -1.],
       [ 0.,  0.,  0.,  0.,  0., -1.,  0.,  1.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0., -1.]])
```

- What do you notice about the last two rows?
- Would this be allowed if `U` were upper echelon?

Part II: Getting to Echelon Form

In [6]:

```
#keep
def swap_rows(mat, i, j):
    temp = mat[i].copy()
    mat[i] = mat[j]
    mat[j] = temp
```

In [7]:

```
#keep
def m_echelon(A):
    m, n = A.shape
    M = np.eye(m)
    U = A.copy()

    row = 0
    for col in range(min(m, n)):
        piv_row = row + np.argmax(np.abs(U[row:, col]))

        if abs(U[piv_row, col]) == 0:
            # column is exactly zero
            continue

        swap_rows(U, row, piv_row)
        swap_rows(M, row, piv_row)

        for el_row in range(row + 1, m):
            fac = -U[el_row, col]/U[row, col]
            U[el_row] += fac*U[row]
            M[el_row] += fac*M[row]

        row += 1

    return M, U
```

Compute M and U, and check that $MA = U$:

In [8]:

```
#keep
M, U = m_echelon(A)

diff = M.dot(A)-U

print(la.norm(diff))
```

```
0.0
```

Let's see if U is actually in echelon form:

In [9]:

```
#keep
U
```

Out[9]:

```
array([[-1.,  0.,  1.,  0.,  0.,  0., -1.,  0.,  0.],
       [ 0., -1.,  1.,  1., -1.,  1., -1.,  0., -1.],
       [ 0.,  0., -1., -1.,  0.,  0.,  0., -1.,  0.],
       [ 0.,  0.,  0.,  0., -1.,  1., -1., -1., -1.],
       [ 0.,  0.,  0.,  0.,  0., -1.,  0.,  1.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.]])
```

And what does M look like?

In [10]:

```
#keep
M
```

Out[10]:

```
array([[ 0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  1.,  1.,  0.,  0.,  0.,  0.],
       [ 1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 1.,  0.,  0.,  1.,  1.,  0.,  0.,  1.,  0.],
       [ 0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.],
       [ 0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 1.,  1.,  0.,  1.,  1.,  1.,  1.,  1.,  1.]])
```

... not much structure here.

But we can still have *something* a little like LU:

In [11]:

```
#keep
A - la.inv(M).dot(U)
```

Out[11]:

```
array([[ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.]])
```

In [ ]:

```

```

In [12]:

```
A = np.random.rand(5,5)
```

In [13]:

```
M, U = m_echelon(A)
```

In [14]:

```
M
```

Out[14]:

```
array([[ 0.        ,  0.        ,  0.        ,  0.        ,  1.
],
       [ 1.        ,  0.        ,  0.        ,  0.        , -0.01735
781],
       [-0.94411529,  0.        ,  1.        ,  0.        , -0.08147
027],
       [-0.15695783,  0.        ,  0.30258661,  1.        , -0.59104
586],
       [-0.66098952,  1.        ,  0.18899847,  0.23650075, -0.90859
199]])
```

In [16]:

```
U.round(3)
```

Out[16]:

```
array([[ 0.777,  0.194,  0.063,  0.318,  0.569],
       [ 0.   ,  0.752,  0.825,  0.592,  0.306],
       [-0.   ,  0.   , -0.73 , -0.034,  0.535],
       [-0.   ,  0.   , -0.   ,  0.65 ,  0.839],
       [-0.   ,  0.   , -0.   ,  0.   ,  0.155]])
```

In [17]:

```
L = np.linalg.inv(M)
```

In [18]:

```
L
```

Out[18]:

```
array([[ 0.01735781,  1.        ,  0.        ,  0.        ,  0.
],
       [ 0.76814609,  0.51299527, -0.11743651, -0.23650075,  1.
],
       [ 0.09785804,  0.94411529,  1.        ,  0.        ,  0.
],
       [ 0.56415977, -0.12871882, -0.30258661,  1.        ,  0.
],
       [ 1.        ,  0.        ,  0.        ,  0.        ,  0.
]])
```

In [19]:

```
L.round(2)
```

Out[19]:

```
array([[ 0.02,  1.  ,  0.  ,  0.  ,  0.  ],
       [ 0.77,  0.51, -0.12, -0.24,  1.  ],
       [ 0.1 ,  0.94,  1.  ,  0.  ,  0.  ],
       [ 0.56, -0.13, -0.3 ,  1.  ,  0.  ],
       [ 1.  ,  0.  ,  0.  ,  0.  ,  0.  ]])
```

In [ ]:

```

```