03-03-Matrices

September 2, 2018

1 Introduction to Matrices

In general terms, a matrix is an array of numbers that are arranged into rows and columns.

1.1 Matrices and Matrix Notation

A matrix arranges numbers into rows and columns, like this:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \tag{1}$$

Note that matrices are generally named as a capital letter. We refer to the *elements* of the matrix using the lower case equivalent with a subscript row and column indicator, like this:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \end{bmatrix} \tag{2}$$

In Python, you can define a matrix as a 2-dimensional *numpy.**array***, like this:

```
In [1]: import numpy as np

        A = np.array([[1,2,3],
                      [4,5,6]])
        print (A)
```

```
[[1 2 3]
 [4 5 6]]
```

You can also use the *numpy.**matrix*** type, which is a specialist subclass of ***array***:

```
In [2]: import numpy as np

        M = np.matrix([[1,2,3],
                       [4,5,6]])
        print (M)
```

```
[[1 2 3]
 [4 5 6]]
```

There are some differences in behavior between ***array*** and ***matrix*** types - particularly with regards to multiplication (which we'll explore later). You can use either, but most experienced Python programmers who need to work with both vectors and matrices tend to prefer the ***array*** type for consistency.

1.2 Matrix Operations

Matrices support common arithmetic operations.

1.2.1 Adding Matrices

To add two matrices of the same size together, just add the corresponding elements in each matrix:

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

In this example, we're adding two matrices (let's call them ***A*** and ***B***). Each matrix has two rows of three columns (so we describe them as 2x3 matrices). Adding these will create a new matrix of the same dimensions with the values a1,1 + b1,1, a1,2 + b1,2, a1,3 + b1,3,a2,1 + b2,1, a2,2 + b2,2, and a2,3 + b2,3. In this instance, each pair of corresponding elements(1 and 6, 2, and 5, 3 and 4, etc.) adds up to 7.

Let's try that with Python:

```
In [3]: import numpy as np

        A = np.array([[1,2,3],
                      [4,5,6]])
        B = np.array([[6,5,4],
                      [3,2,1]])
        print(A + B)
```

```
[[7 7 7]
 [7 7 7]]
```

1.2.2 Subtracting Matrices

Matrix subtraction works similarly to matrix addition:

$$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}-\begin{bmatrix}6 & 5 & 4\\3 & 2 & 1\end{bmatrix}=\begin{bmatrix}-5 & -3 & -1\\1 & 3 & 5\end{bmatrix} \tag{4}$$

Here's the Python code to do this:

```
In [4]: import numpy as np

        A = np.array([[1,2,3],
                      [4,5,6]])
        B = np.array([[6,5,4],
                      [3,2,1]])
        print (A - B)
```

```
[[-5 -3 -1]
 [ 1  3  5]]
```

Conformability In the previous examples, we were able to add and subtract the matrices, because the *operands* (the matrices we are operating on) are ***conformable*** for the specific operation (in this case, addition or subtraction). To be conformable for addition and subtraction, the operands must have the same number of rows and columns. There are different conformability requirements for other operations, such as multiplication; which we'll explore later.

1.2.3 Negative Matrices

The nagative of a matrix, is just a matrix with the sign of each element reversed:

$$C = \begin{bmatrix} -5 & -3 & -1 \\ 1 & 3 & 5 \end{bmatrix} \tag{5}$$

$$-C = \begin{bmatrix} 5 & 3 & 1 \\ -1 & -3 & -5 \end{bmatrix} \tag{6}$$

Let's see that with Python:

```
In [5]: import numpy as np

        C = np.array([[-5,-3,-1],
                      [1,3,5]])
        print (C)
        print (-C)
```

```
[[-5 -3 -1]
 [ 1  3  5]]
[[ 5  3  1]
 [-1 -3 -5]]
```

1.2.4 Matrix Transposition

You can *transpose* a matrix, that is switch the orientation of its rows and columns. You indicate this with a superscript **T**, like this:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} \tag{7}$$

In Python, both *numpy.**array*** and *numpy.**matrix*** have a **T** function:

```
In [6]: import numpy as np

        A = np.array([[1,2,3],
                      [4,5,6]])
        print(A.T)
```

```
[[1 4]
 [2 5]
 [3 6]]
```