NumPy Matrix Multiplication

You’ve heard a lot about matrix multiplication in the last few sections – now you’ll get to see how to do it with NumPy. However, it’s important to know that NumPy supports several types of matrix multiplication.

Element-wise Multiplication

You saw some element-wise multiplication already. You accomplish that with the `multiply` function or the `*` operator. Just to revisit, it would look like this:

```
m = np.array([[1,2,3],[4,5,6]])
m
# displays the following result:
# array([[1, 2, 3],
#        [4, 5, 6]])

n = m * 0.25
n
# displays the following result:
# array([[ 0.25,  0.5 ,  0.75],
#        [ 1.  ,  1.25,  1.5 ]])

m * n
# displays the following result:
# array([[ 0.25,  1.  ,  2.25],
#        [ 4.  ,  6.25,  9.  ]])

np.multiply(m, n)   # equivalent to m * n
# displays the following result:
# array([[ 0.25,  1.  ,  2.25],
#        [ 4.  ,  6.25,  9.  ]])
```

Matrix Product

To find the matrix product, you use NumPy’s `matmul` function.

If you have compatible shapes, then it’s as simple as this:

```
a = np.array([[1,2,3,4],[5,6,7,8]])
a
# displays the following result:
# array([[1, 2, 3, 4],
#        [5, 6, 7, 8]])
a.shape
# displays the following result:
# (2, 4)

b = np.array([[1,2,3],[4,5,6],[7,8,9],[10,11,12]])
b
# displays the following result:
# array([[ 1,  2,  3],
#        [ 4,  5,  6],
#        [ 7,  8,  9],
#        [10, 11, 12]])
b.shape
# displays the following result:
# (4, 3)

c = np.matmul(a, b)
c
# displays the following result:
# array([[ 70,  80,  90],
#        [158, 184, 210]])
c.shape
# displays the following result:
# (2, 3)
```

If your matrices have incompatible shapes, you'll get an error, like the following:

```
np.matmul(b, a)
# displays the following error:
# ValueError: shapes (4,3) and (2,4) not aligned: 3 (dim 1) != 2
(dim 0)
```

NumPy’s `dot` function

You may sometimes see NumPy’s `dot` function in places where you would expect a `matmul`. It turns out that the results of `dot` and `matmul` are the same *if the matrices are two dimensional*.

So these two results are equivalent:

```
a = np.array([[1,2],[3,4]])
a
# displays the following result:
# array([[1, 2],
#        [3, 4]])

np.dot(a,a)
# displays the following result:
# array([[ 7, 10],
#        [15, 22]])

a.dot(a)  # you can call `dot` directly on the `ndarray`
# displays the following result:
# array([[ 7, 10],
#        [15, 22]])

np.matmul(a,a)
# array([[ 7, 10],
#        [15, 22]])
```

While these functions return the same results for two dimensional data, you should be careful about which you choose when working with other data shapes. You can read more about the differences, and find links to other NumPy functions, in the `matmul` and `dot` documentation.