6_ndarray_operations

May 16, 2019

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

Arrary operations

```
In [ ]: test_a = np.array([[1,2,3],[4,5,6]], float)
        test_a

In [ ]: test_a + test_a # Matrix + Matrix

In [ ]: test_a - test_a # Matrix - Matrix

In [ ]: test_a * test_a # Matrix element

In [ ]: matrix_a = np.arange(1,13).reshape(3,4)
        matrix_a * matrix_a
```

dot product

```
In [ ]: test_a = np.arange(1,7).reshape(2,3)
        test_b = np.arange(7,13).reshape(3,2)

In [ ]: test_a

In [ ]: test_b

In [ ]: test_a.dot(test_b)

In [ ]: test_a = np.arange(1,7).reshape(2,3)
        test_a

In [ ]: test_a.transpose()

In [ ]: test_a.T

In [ ]: test_a.T.dot(test_a) # Matrix

In [ ]: test_a.dot(test_a.T)
```

broadcasting

```
In [ ]: test_matrix = np.array([[1,2,3],[4,5,6]], float)
        scalar = 3

In [ ]: test_matrix + scalar # Matrix - Scalar

In [ ]: test_matrix - scalar # Matrix - Scalar

In [ ]: test_matrix * 5 # Matrix - Scalar

In [ ]: test_matrix / 5 # Matrix - Scalar

In [ ]: test_matrix // 0.2 # Matrix - Scalar

In [ ]: test_matrix ** 2 # Matrix - Scalar

In [ ]: test_matrix = np.arange(1,13).reshape(4,3)
        test_vector = np.arange(10,40,10)
        print(test_matrix, test_vector)

        test_matrix+ test_vector
```

numpy performance

```
In [ ]: def sclar_vector_product(scalar, vector):
            result = []
            for value in vector:
                result.append(scalar * value)
            return result

        iternation_max = 100000000

        vector = list(range(iternation_max))
        scalar = 2

        %timeit sclar_vector_product(scalar, vector) # for loop

In [ ]: %timeit [scalar * value for value in range(iternation_max)] # list comprehension
        %timeit np.arange(iternation_max) * scalar # numpy
```