Content

- **Reshape with -ve index**
- **Matrix Multiplication**
  - `matmul()` , `@` , `dot()`
- **Vectorization**
  - `np.vectorize()`

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

Reshape in 2D array

We saw reshape and flatten. What if i want to convert a matrix to 1D array using `reshape()`

Question: What should I pass in `A.reshape()` if I want to use it to convert `A` to 1D vector?

- **(1, 1)? - NO**
- It means we only have a single element
- But **we don't have a single element**

In [ ]:
```
1 A = np.arange(12).reshape(3,4)
```

In [ ]:
```
1 A.reshape(1, 1)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-138-902e5c35e0d3> in <module>()
----> 1 A.reshape(1, 1)

ValueError: cannot reshape array of size 12 into shape (1,1)
```

- So, **(1, 12)? - NO**
- It will **still remain a 2D Matrix with dimensions** 1×12

In [ ]:
```
1 A.reshape(1, 12)
```

Out[139]:
```
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11]])
```

- We need a vector of dimension (12,)
- So we need to pass only 1 dimension in `reshape()`

In [ ]:
```
1 A.reshape(12)
```

Out[140]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

So, Be careful while using `reshape()` to convert a Matrix into a 1D vector

What will happen if we pass a negative integer in `reshape()` ?

In [ ]:
```
1 A.reshape(6, -1)
```

Out[141]:
```
array([[ 0,  1],
       [ 2,  3],
       [ 4,  5],
       [ 6,  7],
       [ 8,  9],
       [10, 11]])
```

Surprisingly, it did not give an error

- It is able to **figure out on its own** what should be the **value in-place of negative integer**
- Since **no. of elements in our matrix is 12**
- And **we passed 6 as no. of rows**
- It is **able to figure out** that **no. of columns should be 2**

Same thing happens with this:

In [ ]:
```
1 A.reshape(-1, 6)
```

Out[142]:
```
array([[ 0,  1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10, 11]])
```

Matrix multiplication

Question: What will be output of following?

In [ ]:
```
a = np.arange(5)
b = np.ones(5) * 2
```

In [ ]:
```
a * b
```

Out[6]:
```
array([0., 2., 4., 6., 8.])
```

Recall that, if a and b are 1D, * operation will perform elementwise multiplication

Lets try * with 2D arrays

In [ ]:
```
A = np.arange(12).reshape(3, 4)
A
```

Out[7]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [ ]:
```
B = np.arange(12).reshape(3, 4)
B
```

Out[8]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [ ]:
```
A * B
```

Out[9]:
```
array([[  0,   1,   4,   9],
       [ 16,  25,  36,  49],
       [ 64,  81, 100, 121]])
```

Again did element-wise multiplication

For actual Matrix Multiplication, We have a different method/operator

`np.matmul()`

What is the requirement of dimensions of 2 matrices for Matrix Multiplication?

- **Columns of A = Rows of B** (A **Must condition** for Matric Multiplication)
- **If A is** 3×4**, B can be** $4 \times 3 \ldots$ or $4 \times (SomethingElse)$

So, lets reshape B to 4×3 **instead**

In [ ]:
```
B = B.reshape(4, 3)
B
```

Out[10]:
```
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])
```

In [ ]:
```
np.matmul(A, B)
```

Out[12]:
```
array([[ 42,  48,  54],
       [114, 136, 158],
       [186, 224, 262]])
```

- **We are getting a 3×3 matrix as output**
- So, this is doing Matrix Multiplication

There's a direct operator as well for Matrix Multiplication

`@`

In [ ]:
```
A @ B
```

Out[13]:
```
array([[ 42,  48,  54],
       [114, 136, 158],
       [186, 224, 262]])
```

Question: What will be the dimensions of Matrix Multiplication `B @ A` ?

- 4×4

In [ ]:
```
B @ A
```

Out[14]:
```
array([[ 20,  23,  26,  29],
       [ 56,  68,  80,  92],
       [ 92, 113, 134, 155],
       [128, 158, 188, 218]])
```

There is another method in np for doing Matrix Multiplication

In [ ]:
```
np.dot(A, B)
```

Out[15]:
```
array([[ 42,  48,  54],
       [114, 136, 158],
       [186, 224, 262]])
```

Other cases of `np.dot()`

- It performs dot product when both inputs are 1D array
- It performs multiplication when both input are scalers.

In [ ]:
```
a= np.array([1,2,3])
b = np.array([1,1,1])

```

In [ ]:
```
np.dot(a,b) # 1*1 + 2*1 + 3*1 = 6
```

Out[17]:
```
6
```

In [ ]:
```
np.dot(4,5)
```

Out[18]:
```
20
```

Now, Let's try multiplication of a mix of matrices and vectors

In [ ]:
```
A = np.arange(12).reshape(3, 4)  # A is a 3x4 Matrix
A
```

Out[19]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [ ]:
```
a = np.array([1, 2, 3])  # a although a (3,) can be thought of as row vec
print(a.shape)
```

```
[1 2 3]
(3,)
```

In [ ]:
```
np.matmul(A, a)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-25-76efef6bd8e9> in <module>()
----> 1 np.matmul(A, a)

ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0,
with gufunc signature (n?,k),(k,m?)->(n?,m?) (size 3 is different from 4)
```

Columns of `A` $\neq$ Rows of `a`

Lets try reevrse

In [ ]:
```
1 np.matmul(a, A)
```

Out[26]:
```
array([32, 38, 44, 50])
```

YES, **Columns of `a` (3) = Rows of `A` (3)**