Theory: Operations with several arrays

20 minutes 0 / 5 problems solved Skip this topic Start practicing

178 users solved this topic. Latest completion was about 6 hours ago.

NumPy provides a great variety of ways to operate on arrays. In this topic, we are going to cover the main procedures and discuss their peculiarities.

§1. Basic arithmetic operations on arrays

First of all, do not forget to import the module:

```
import numpy as np
```

The first operations we are going to cover are basic mathematical operations: addition, subtraction, multiplication, division, and exponentiation. They all are element-wise; this means that the operations are applied to pairs of corresponding elements in two arrays and the results are placed in a new array.

1. Addition
   To calculate the sum of two arrays, you can use the `np.add()` function.

```
a1 = np.array([1, 2, 3])
a2 = np.array([4, 5, 6])
a3 = np.add(a1, a2)
print(a3)  # [5 7 9]
```

There is also an opportunity to add two-dimensional arrays or arrays with more than two dimensions. Another way of using addition is `a1 + a2`.

```
a1 = np.array([[1, 2], [3, 4]])
a2 = np.array([[5, 6], [7, 8]])
a3 = a1 + a2
print(a3)
# [[ 6  8]
#  [10 12]]
```

Now mind another example:

```
a1 = np.array([[1, 2], [3, 4]])
a2 = np.array([[5, 6, 7], [8, 9, 10]])
a3 = np.add(a1, a2)

# ValueError: operands could not be broadcast together with shapes (2,2) (2,3)
```

The thing is, the first array has two columns, while the second one has three columns, so the `ValueError` is raised because of the incompatibility of different shapes.

Now let's analyze another example.

```
a1 = np.array([[1, 2], [3, 4]])
a2 = np.array([5, 6])
a3 = np.add(a1, a2)
print(a3)
# [[ 6  8]
#  [ 8 10]]
```

The operands have different shapes, but there are no errors. So, what happens in this case? The shape of the first operand is `(2,2)`, the shape of the second one is `(2,)`. To make them compatible and apply an element-wise addition, NumPy uses **broadcasting.** Broadcasting can be described as a set of rules for applying element-wise operations on arrays of different shapes. When you work with two arrays, NumPy always compares their sizes and turns them into one shape, *if possible*.

Current topic:

Operations with several arrays

Topic depends on:

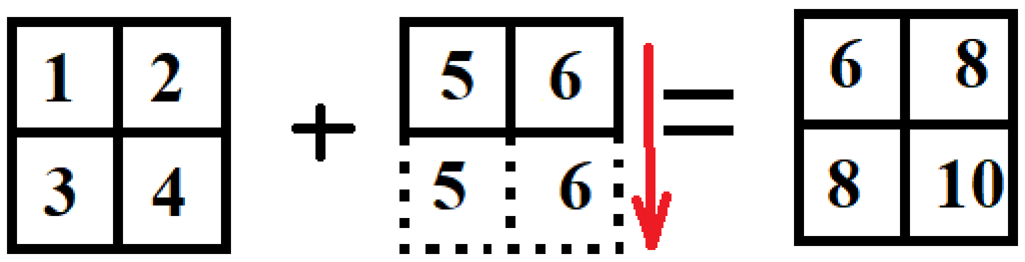
- Matrix Multiplication
- Intro to NumPy

Topic is required for:

- Boolean operations on arrays
- Submodule linalg

Table of contents:

In the example, broadcasting goes as follows: as the last dimensions have an equal number of elements, it takes the second operand and expands it to a larger size:



In other words, the broadcasting procedure replicates `[5 6]` and adds it to `[1 2]` and `[3 4]`. The resultant array has the `(2,2)` shape.

There is another example for you:

```
a1 = np.array([[[1, 2, 3]], [[4, 5, 6]]])
a2 = np.array([[7], [8]])
a3 = np.add(a1, a2)
print(a3)
# [[[ 8  9 10]
#   [ 9 10 11]]
#
#  [[11 12 13]
#   [12 13 14]]]
```

The shape of the first operand is `(2,1,3)`, the second one has the `(2,1)` shape. What does broadcasting do in this case? Dimensions with the size equal to 1 are expanded to match dimensions in the first array: `7` is replicated, so it can be added to `[1 2 3]`; `8` is also replicated and added to `[1 2 3]`. Then, `7` is added to `[4 5 6]`, and so is `8`. The shape of the resultant array is `(2,2,3)`.

At the end of the day, broadcasting tries to match different sizes so that arithmetic operations can be performed. Most of the examples below will include broadcasting, too, so that you can understand how it works better. You can read more about broadcasting in the official Numpy guidelines.

2. Subtraction

   To subtract arrays, we can use `np.subtract()`:

```
a1 = np.array([[2, 7], [8, 13]])
a2 = np.array([5])
a3 = np.subtract(a1, a2)
print(a3)
# [[-3  2]
#  [ 3  8]]
```

   In this case, the only element of the second array is also replicated. Mind that you can type `a1 - a2` instead of `np.subtract(a1, a2)`. Moreover, you can subtract the integer `5` instead of the NumPy array `[5]`, the result will stay the same.

3. Multiplication

   `np.multiply()` will multiply arguments of arrays element-wise:

```
a1 = np.array([[[1, 2, 3], [3, 4, 5]], [[5, 6, 7] ,[7, 8, 9]]])
a2 = np.array([[-1, 0, 7], [12, 14, 5]])
a3 = np.multiply(a1, a2)
print(a3)
# [[[ -1   0  21]
#   [ 36  56  25]]
#
#  [[ -5   0  49]
#   [ 84 112  45]]]
```

   As you can see, the shape of the resultant array is equal to the shape of the bigger array. Again, writing `a1 * a2` is equal to `np.multiply(a1, a2)`.

4. Division

`np.divide()` allows us to divide two arrays:

```
a1 = np.array([[[-6, 12, 3]], [[2, 15, 3]]])
a2 = np.array([[2], [4]])
a3 = np.divide(a1, a2)
print(a3)
# [[[-3.    6.    1.5 ]
#   [-1.5   3.    0.75]]
#
#  [[ 1.    7.5   1.5 ]
#   [ 0.5   3.75  0.75]]]
```

You can also use `a1 / a2` instead of `np.divide(a1, a2)`.

5. Exponentiation

   `np.power()` raises elements of the first array to powers from the second array. It is also an element-wise operation.

```
a1 = np.array([[1, 2], [3, 4]])
a2 = np.array([[5, 6], [7, 8]])
a3 = np.power(a1, a2)
print(a3)
# [[    1    64]
#  [ 2187 65536]]
```

   `a1 ** a2` is another way to do so.

To sum up, NumPy allows us to implement basic arithmetic operations on several arrays. If array shapes are different, sometimes it is still possible to perform operations on them, with the help of broadcasting.

§2. Array multiplication vs matrix product

When discussing multiplication in NumPy, we should distinguish two different operations: `np.multiply()` and `np.matmul()`. Let's take a look at the example:

```
a1 = np.array([[1, 2], [5, 8]])
a2 = np.array([[4, 6], [7, 3]])

a3 = np.multiply(a1, a2)
print(a3)
# [[ 4 12]
#  [35 24]]

a4 = np.matmul(a1, a2)

print(a4)

# [[18 12]

#  [76 54]]
```

As far as we know, `np.multiply()` is applied when we want to compute element-wise multiplication. By contrast, `np.matmul()` is used for obtaining **the matrix product** of two arrays, it is not an element-wise operation. You can also change `np.matmul(a1, a2)` to `a1 @ a2` which is the same way to obtain the matrix product.

§3. Array transposing

Finally, it is worth mentioning that NumPy provides a way for transposing a matrix using `np.transpose()` or the alternative `array.T`:

```
a1 = np.array([[1, 2], [3, 4]])
a2 = np.transpose(a1)
print(a2)
# [[1 3]
#  [2 4]]

a3 = a1.T
print(a3 == a2)  # True
```

It can also be implemented if the number of rows and the number of columns are not equal:

```
a1 = np.array([[1, 2], [3, 4], [5, 6]])
a2 = a1.T
print(a2)
# [[1 3 5]
#  [2 4 6]]
```

However, if you deal with a one-dimensional array, the same array will be returned:

```
a1 = np.array([1, 2, 3, 4])
a2 = np.transpose(a1)
print(a2)
# [1 2 3 4]
```

§4. Conclusion

In this topic, we have learned:

- how to use basic arithmetic procedures for two arrays;
- the basics of broadcasting in NumPy;
- the difference between array multiplication and matrix product;
- how to transpose arrays.

Now it is time for you to practice!

Report a typo

17 users liked this theory. 0 didn't like it. **What about you?**

Start practicing

Comments (0) Hints (0) Useful links (0) Show discussion