NumPy Broadcasting

Summary: in this tutorial, you'll learn about NumPy broadcasting and understand how it works.

Introduction to the NumPy broadcasting

In previous tutorials, you learned how to perform arithmetic operations on equal-sized arrays using the `add()` (https://www.pythontutorial.net/python-numpy/numpy-add/), `subtract()` (https://www.pythontutorial.net/python-numpy/numpy-subtract/), `multiply()` (https://www.pythontutorial.net/python-numpy/numpy-multiply/), and `divide()` (https://www.pythontutorial.net/python-numpy/numpy-divide/) functions or as `+`, `-`, `*`, and `/` operators.

To perform arithmetic operations on arrays of different shapes, NumPy uses a technique called broadcasting.

By definition, broadcasting is a set of rules for applying arithmetic operations on arrays of different shapes. We'll cover these rules in detail shortly.

Before that, let's take some simple broadcasting examples. For instance, you can use the + operator to add a number to an array like this:

```
import numpy as np

a = np.array([1, 2, 3])
b = a + 1
print(b)
```

Output:

```
[2 3 4]
```

This example adds the number one to a 1D array using the operator `+`. Internally, NumPy adds the number 1 to every element of the array. This technique is called broadcasting.

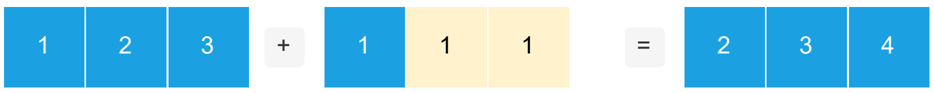
In other words, NumPy broadcasts the number one across the first dimension to match the shape of the 1D array.

Conceptually, you can think of the above broadcasting as equivalent to the following:

```
import numpy as np

a = np.array([1, 2, 3])
b = a + np.array([1, 1, 1])
print(b)
```

So adding the number one to a 1D array is like duplicating the number one into another 1D array [1, 1, 1] and adding that array to the array:



However, NumPy broadcasts number one without duplicating it. By doing this, NumPy can manage effciently and speeds up calculation in most cases.

Similarly, you can add a 1D array to a 2D array using broadcasting like this:

```
import numpy as np

```

```
a = np.array([
    [1, 2, 3],
    [4, 5, 6]
])
b = np.array([10, 20, 30])
c = a + b
print(c)
```

Output:

```
[[11 22 33]
 [14 25 36]]
```

In this example, NumPy broadcasts the 1D array `b` across the second dimension to match the shape of the array a.

NumPy broadcasting rules

NumPy defines a set of rules for broadcasting:

- Rule 1: if two arrays have different dimensions, it pads ones on the left side of the shape of the array that has fewer dimensions.
- Rule 2: if two dimensions of arrays do not match in any dimension, the array with a shape equal to one in that dimension is stretched (or broadcast) to match the shape of another array.
- Rule 3: if any dimension of two arrays is not equal and neither is equal to one, NumPy raises an error.

Let's take some examples to understand these rules.

1) NumPy broadcasting on one array example

The following example adds a 2-D array to a 1D array:

```
import numpy as np

a = np.array([
    [1, 2, 3],
    [4, 5, 6]
])
b = np.ones(3)
c = a + b
print(c)
```

Output:

```
[[2. 3. 4.]
 [5. 6. 7.]]
```

The following table shows the shapes of a and b:

Array	Shape
a	(2,3)
b	(3,)

By rule 1, because the array b has fewer dimensions, NumPy pads one on the left:

Array	Shape
a	(2,3)
b	(1,3)

By rule 2, the first dimensions of two shapes are not equal, NumPy stretches (or broadcasts) the first dimension of the array b to match:

Array	Shape
a	(2,3)
b	(2,3)

Now, the dimensions of both arrays match. The shape of the result array is (2,3).

2) NumPy broadcasting on both arrays example

The following example illustrates the case where NumPy broadcasts both arrays:

```
import numpy as np

a = np.array([
    [1],
    [2],
    [3],
])
print(f"a shape: ", a.shape)

b = np.array([1, 2, 3])
print(f"b shape: ", b.shape)

c = a + b
print(c)
print(f"c shape: ", c.shape)
```

Output:

```
a shape:  (3, 1)
b shape:  (3,)
[[2 3 4]
 [3 4 5]
 [4 5 6]]
c shape:  (3, 3)
```

In this example, the shape of a and b arrays are (3,1) and (3,) respectively.

Array	Shape
a	(3,1)

Array	Shape
b	(3,)

By rule 1, NumPy pads the shape of b with ones:

Array	Shape
a	(3,1)
b	(1,3)

By rule 2, NumPy stretches the dimensions of both arrays a and b to match because they're both ones:

Array	Shape
a	(3,3)
b	(3,3)

The resulting array has the shape of (3,3).

3) NumPy broadcasting with error example

The following example adds two arrays that are not compatible:

```
import numpy as np

a = np.array([
    [1, 2],
    [3, 4],
    [5, 6],
])
print(f"a shape: ", a.shape)

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

```
print(f"b shape: ", b.shape)

c = a + b
```

It issues the following error:

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

In this example, the array a and b have the following shapes:

Array	Shape
a	(3,2)
b	(3,)

By rule 1, NumPy pads the shape of the second array with ones:

Array	Shape
a	(3,2)
b	(1,3)

By rule 2, NumPy stretches the first dimension of the b array from 1 to 3 to match:

Array	Shape
a	(3,2)
b	(3,3)

By rule 3, the final shapes do not match, therefore, NumPy raises an error.

Summary

- NumPy broadcasting is a set of rules for applying arithmetic operations on arrays of different shapes.