Effective Programming Practices for Economists

Scientific Computing

Broadcasting

Janoś Gabler and Hans-Martin von Gaudecker

Examples of broadcasting

```
>>> a = np.zeros((2, 3), dtype=np.int64)
>>> a
array([[0, 0, 0],
       [0, 0, 0]])

# element-wise addition
>>> a + 1
array([[1, 1, 1],
       [1, 1, 1]])

# row-wise addition
>>> a + np.array([1, 2, 3])
array([[1, 2, 3],
       [1, 2, 3]])

# column-wise addition
>>> a + np.array([[4], [5]])
array([[4, 4, 4],
       [5, 5, 5]])
```

- Arrays don’t have to have identical shapes to do calculations between them
- Smaller arrays are broadcasted to the larger shape
- Shapes need to be compatible as defined by the broadcasting rules

The broadcasting rule

Two arrays are compatible for broadcasting if for each *trailing dimension* (i.e., starting from the end) the axis lengths match or if either of the lengths is 1. Broadcasting is then applied over the missing or length 1 dimensions

More information and examples in the documentation

Detailed walk through examples

```
>>> a = np.zeros((2, 3), dtype=np.int64)

# row-wise addition
>>> b = np.array([1, 2, 3])
>>> b.shape
(3,)
>>> a + b
array([[1, 2, 3],
       [1, 2, 3]])

# column-wise addition
>>> c = np.array([[4], [5]])
>>> c.shape
(2, 1)
>>> a + c
array([[4, 4, 4],
       [5, 5, 5]])
```

- `a + b` : Axis 1 matches (3), axis 0 is broadcasted twice
- `a + c` : Axis 1 is has length 1, axis 0 matches (2), axis 1 is broadcasted three times
- More efficient than repeating arrays!

Advanced example: Outer product

```
>>> a = np.array([1, 2, 3])
>>> b = np.array([4, 5, 6])
>>> np.outer(a, b)
array([[ 4,  5,  6],
       [ 8, 10, 12],
       [12, 15, 18]])

>>> a.reshape(1, 3) * b.reshape(3, 1)
array([[ 4,  8, 12],
       [ 5, 10, 15],
       [ 6, 12, 18]])

>>> a * b.reshape(3, 1)
array([[ 4,  5,  6],
       [ 8, 10, 12],
       [12, 15, 18]])
```

- Here, broadcasting is used to calculate an outer product without using the `np.outer` function
- `a` is reshaped to a row vector
- `b` is reshaped to a column vector
- Broadcasting rules apply along both axes!
- `a` would be implicitly treated as a row vector, too.