Introduction to NumPy

Moussa Doumbia, Ph.D.

Howard University

January 31, 2024

Table of Contents

Introduction to NumPy

- NumPy is a fundamental package for scientific computing in Python.
- It provides support for large, multi-dimensional arrays and matrices.
- NumPy offers a wide range of mathematical functions to operate on these arrays.
- Key features: powerful N-dimensional array object, sophisticated functions, tools for integrating C/C++ and Fortran code, linear algebra, Fourier transform, and random number capabilities.

Import Libraries

```
import numpy as np
```

- The above code imports the NumPy library and aliases it as ‘np‘.
- This alias is a standard convention used in the Python community.

Basics of NumPy Arrays

- Arrays in NumPy are n-dimensional, homogeneous, and are indexed by a tuple of nonnegative integers.
- The number of dimensions is the rank of the array.
- The shape of an array is a tuple of integers giving the size of the array along each dimension.

```
# Create a 1D array
a = np.array([1, 2, 3])

# Create a 2D array
b = np.array([[1, 2, 3], [4, 5, 6]])
```

Array Indexing

- NumPy offers several ways to index into arrays.
- Slicing: Similar to Python lists, NumPy arrays can be sliced.
- Integer array indexing: NumPy arrays can be indexed with other arrays or any other sequence with integers.

```
# Slicing
a = np.array([1, 2, 3, 4, 5])
a[1:3]  # Output will be [2, 3]

# Integer array indexing
a[[1, 3, 4]]  # Output will be [2, 4, 5]
```

NumPy Array Operations

- NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data.
- Operations on NumPy arrays are very fast as they use optimized C API.

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

# Elementwise sum
c = a + b  # Output will be [6, 8, 10, 12]

# Elementwise product
d = a * b  # Output will be [5, 12, 21, 32]
```

Array Concatenation and Splitting

- Concatenation of arrays is the operation of joining two or more arrays.
- Splitting is the operation of dividing a single array into multiple.

```
# Concatenation
x = np.array([1, 2, 3])
y = np.array([4, 5, 6])
np.concatenate([x, y])  # Output will be [1, 2, 3, 4, 5, 6]

# Splitting
x = np.array([1, 2, 3, 4, 5, 6])
np.split(x, 2)  # Output will be [array([1, 2, 3]), array([4,
```

Array Manipulation

- NumPy provides various functions to change the shape of arrays, stack arrays, or split arrays.
- reshape, flatten, ravel, and transpose are some of the common functions used for array manipulation.

```
a = np.array([[1, 2, 3], [4, 5, 6]])

# Reshape
np.reshape(a, (3, 2))

# Flatten
a.flatten()

# Transpose
a.T
```

Saving/Loading Notebooks

- NumPy allows you to save and load arrays to and from disk.
- Functions like np.save, np.load, np.savetxt, and np.loadtxt are used for this purpose.

```python
a = np.array([1, 2, 3, 4, 5])

# Save to .npy file
np.save('my_array', a)

# Load from .npy file
b = np.load('my_array.npy')

# Save to .txt file
np.savetxt('my_array.txt', a)

# Load from .txt file
c = np.loadtxt('my_array.txt')
```