Tutorial 2: Introduction to Numpy

EE2405

嵌入式系統與實驗

Embedded System Lab

A Motivating Example (1/2)

```
import numpy as np
import timeit

A=np.random.random((10,10))
B=np.random.random((10,10))
C=np.zeros((10,10))
A_l=A.tolist()
B_l=B.tolist()
C_l=C.tolist()

t1=timeit.timeit('C=np.matmul(A, B)', number=100, globals=globals())
print("Numpy Execution time = ", "{0:.6f}".format(t1), "seconds.")
```

A Motivating Example (2/2)

```
def Matrix_Multiply(X, Y, result):
    for i in range(len(X)):
        for j in range(len(Y[0])):
            for k in range(len(Y)):
                result[i][j] += X[i][k] * Y[k][j]

t2=timeit.timeit('Matrix_Multiply(A_l, B_l, C_l)', number=100,
globals=globals())

print("Python Execution time = ", "{0:.6f}".format(t2),
"seconds.")
print("Numpy/Python speedup ratio =", "{0:.2f}".format(t2/t1))
```

Output of Example

```
$ python3 numpy1.py
Numpy Execution time =  0.000242 seconds.
Python Execution time =  0.022110 seconds.
Numpy/Python speedup ratio = 91.43
```

Numpy Introduction

- Numerical Python
- Features:
  - N-dimensional array object
  - Mathematical and logical operations on arrays.
  - Fourier transforms
  - Linear algebra
  - Random numbers

Overview

- Arrays
- Shaping and transposition
- Mathematical Operations
- Indexing and slicing
- Broadcasting

Numpy Arrays

- Contiguous one-dimensional segment of computer memory + Index

```python
import numpy as np
a = np.array([[1,2,3],[4,5,6]],dtype=np.float32)
print(a.ndim, a.shape, a.dtype)
```

```
2 (2, 3) float32
```

- Array
    - Arrays can have any number of dimensions, including zero (a scalar).
    - Arrays are typed: np.uint8, np.int64, np.float32, np.float64
    - Arrays are dense. Each element of the array exists and has the same type.

Array Creation API

- np.ones, np.zeros
  - Return a new array of ones/zeros in a given shape and type.
- np.arrange
  - Return a sequence of values in a range of a given step
- np.astype
  - Copy of the array, cast to a specified type.
- np.zeros_like, np.ones_like
  - Return an array of zeros/ones with the same share and type of a given array
- np.random.random
  - Return an array of random floats in [0.0, 1.0).
- np.concatenate
  - Join a sequence of arrays along an existing axis.

Examples (1/2)

- `np.zeros((2, 2), dtype=np.float32)`

```
[[0. 0.]
 [0. 0.]]
```

- `np.arange(3,7)`

```
[3 4 5 6]
```

- `np.arange(3,7,2)`

```
[3 5]
```

- `a = np.array([[1.1,2.2,3.3],[4.4,5.5,6.6]])`
- `a.astype(np.uint16)`

```
[[1 2 3]
 [4 5 6]]
```

Examples (2/2)

- `np.ones_like(a)`

```
[[1 1 1]
 [1 1 1]]
```

- `a = np.array([[1, 2], [3, 4]])`
- `b = np.array([[5, 6]])`
- `np.concatenate((a, b), axis=0)`

```
[[1 2]
 [3 4]
 [5 6]]
```

Array Shaping

- Total number of elements cannot change.
- Use -1 to infer axis shape
- Row-major by default (MATLAB is column-major)

Examples of Reshaping

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

```
[[1 2]
 [3 4]
 [5 6]]
```

- `a = a.reshape(2,-1)`

```
[[1 2 3]
 [4 5 6]]
```

- `a = a.ravel()`

```
[1 2 3 4 5 6]
```

Transposition

- `a = np.arange(10).reshape(5,2)`
- `a.T`

```
[[0 1]
 [2 3]
 [4 5]
 [6 7]
 [8 9]]

[[0 2 4 6 8]
 [1 3 5 7 9]]
```

- np.transpose permutes axes.
- a.T transposes the first two axes.

View or Copy

- Views share data with the original array, like references in C++.
  - Altering entries of a view, changes the same entries in the original.
- Please refer to Numpy document to know whether a function returns a view or a copy.
- Np.copy, np.view make explicit copies and views.

Saving to a File

```
import numpy as np
from tempfile import TemporaryFile
outfile = TemporaryFile()
x = np.arange(10)
y = np.sin(x)
np.savez(outfile, x=x, y=y)
npzfile = np.load(outfile)
x_read=npzfile['x']
```

Mathematical operators

- Arithmetic operations are element-wise
- Logical operator return a bool array
- In place operations modify the array

Element-wise Operations

- Dot product example
- `a=np.arange(1,4)`
- `b=np.arange(4,7)`
- `c=a*b`

```
[1 2 3]
[4 5 6]
[ 4 10 18]
```

Universal Functions (ufuncs)

- Element-wise
- Examples:
  - np.exp
  - np.sqrt
  - np.sin
  - np.cos
  - np.isnan

- `a*=np.pi*2`
- `b=np.sin(a)`

```
[[6.1  0.83 0.86]
 [0.71 5.1  4.62]
 [4.02 1.03 3.35]]

[[-0.19  0.74  0.76]
 [ 0.65 -0.92 -1.  ]
 [-0.77  0.86 -0.21]]
```

Logical Operations on Array

- `a=np.random.random((3,3))`
- `b=a>0.5`

```
[[0.04 0.   0.01]
 [0.81 0.11 0.6 ]
 [0.96 0.6  0.43]]

[[False False False]
 [ True False  True]
 [ True  True False]]
```

Array Indexing Examples

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

- `print(a[0,0])`

```
1
```

- `print(a[1,0])`

```
4
```

- `print(a[2,0])`

```
7
```

- `print(a[0,-1])`

```
3
```

- `print(a[-1,0])`

```
7
```

- `print(a[-1,-1])`

```
9
```

Array Slicing Examples

- “start:stop:step” can be used to slice an array

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

- `print(a[0, 1:-1])`

```
[2]
```

- `print(a[0,:])`

```
[1 2 3]
```

- `print(a[0,::-1])`

```
[3 2 1]
```

- `print(a[0,::2])`

```
[1 3]
```

- `print(a[:,0])`

```
[1 4 7]
```

- `print(a[0:-1, 0:-1])`

```
[[1 2]
 [4 5]]
```

Slice Write

- Slices are **views**.
- Writing to a slice overwrites the original array.

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

- `a[1, 1]=-1`

```
[[ 1  2  3]
 [ 4 -1  6]
 [ 7  8  9]]
```

- `a[0:-1, 0:-1]=0`

```
[[0 0 3]
 [0 0 6]
 [7 8 9]]
```

Index by a Boolean Array

```
a=np.arange(1,10).reshape(3,3)
b=np.random.random((3,3))
print(a)
a[b>0.5]=-1
print(b>0.5)
print(a)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]

[[ True  True False]
 [ True False False]
 [False False  True]]

[[-1 -1  3]
 [-1  5  6]
 [ 7  8 -1]]
```