3. Deep learning tools

3.2 Numpy

Manel Martínez-Ramón
Meenu Ajith
Aswathy Rajendra Kurup

Introduction

- Standard package for scientific computing.
- provides advanced mathematical computations and operations using multi-dimensional arrays and matrices.
- The base variable is an multinimensional array (ndarray)
- This lesson summarizes array initialization, types of operations, extracting shape, axis properties.

Arrays

The numpy array can be initialized with the command **np.array**. The array has the following methods:

- **array.shape**: This gives the shape of the array.
- **array.size** : This gives the total number of elements in an array.
- **array.ndim** : Number of axes in an array.
- **array.dtype**: Gives the datatype of elements in the array.

```
1 import numpy as np
2 my_array = np.array([[1, 2, 3, 4],
3                      [5, 6, 7, 8]])
4 #initializing a different data type array
5 my_array.shape #Ouputs (2,4)
```

Predefined arrays

```
1 mat1 = np.zeros((5,4))        #matrix of zeros
2 mat2 = np.ones((3,2))         #matrix of ones
3 mat3 = np.empty((2,2))        #empty matrix
4 mat4 = np.eye(3)              #3x3 identity matrix
5 mat5 = np.full((3,3),2)       #matrix of 2's
6 mat6 = np.arange(1,50,3)      #Sequence of integers
7 mat7 = np.linspace(1,49,17) #The same sequence but with reals
```

Lines 5 and 6 generate a sequence of 17 numbers from 1 to 49.

Changing the shape

```
1 import numpy as np
2 ar = np.array([[1,2],[3,4],[5,6],[7,8])
3 print(ar)
```

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

```
1 ar1 = ar.reshape((2,4)) #modifying the shape
2 print(ar1)
```

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

```
1 ar2=ar1.ravel()
2 print(ar2)
```

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

Stacking and splitting

```
1 array1 = np.array([1,2,3,4]) # initializing two arrays
2 array2 = np.array([5,6,7,8])
3 ar_row = np.hstack((array1,array2)) # using hstack
4 print(ar_row)
```

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

```
1 ar_column = np.column_stack((array1,array2)) # using
      column_stack
2 print(ar_column)
```

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

Stacking and splitting

```
1 print(np.hsplit(ar_row,2)) #splitting along column
```

```
[array([[1],
        [2],
        [3],
        [4]]),
 array([[5],
        [6],
        [7],
        [8]])]
```

```
1 print(np.array_split(ar_row,2, axis = 0)) # along axis = 0
```

```
[array([[1, 5],
        [2, 6]]),
 array([[3, 7],
        [4, 8]])]
```

Arithmetic operations

```
a = np.array([1,2,3,4])       #creating arrays a and b
b = np.array([4.,5.,1.,2.]) #b contains reals
add_ab = np.add(a,b)          #addition
rec_b = np.reciprocal(b)      #reciprocal of array b (inverse)
pow_ab = np.power(a,b)        #power of a to b
sqrt_a = np.sqrt(a)           #square root of a
```

```
zip_obj = zip(a, b)           #create a list of tuples
comp = []                     #create an empty list
for x,y in zip_obj:           #obtain each element zup_obj
    c = complex(x,y)          #compute x+jy
    comp.append(c)            #append these to the empty list
print(list(comp))                # printing complex array
print(list(np.real(comp)))       # getting the real part
print(list(np.imag(comp)))       # getting imag part
```

```
[(1+4j), (2+5j), (3+1j), (4+2j)]
[1.0, 2.0, 3.0, 4.0]
[4.0, 5.0, 1.0, 2.0]
```

Mathematical functions

```
1 a = np.array([30,45,60,90])    #an array of degrees
2 print(np.sin(np.radians(a)))  #convert to rads, print its sine
3 b = np.array([0.35066070245,2.67822320434]) #an array of ints
4 print(np.around(b, 4)) #print their round to the 4th decimal
5 print(np.floor(b))      # use floor command
6 print(np.ceil(b))       # use ceil command
```