Algorithms and Data Structures

Python Packages for Data Science

Agenda

- What are Python packages
- Packages specifically for Data Science
  - Numpy
  - Pandas
  - Matplotlib
  - Scikit-learn

Numpy

- Numpy is the core library for scientific computing.
- Numpy provides a
  - high-performance
  - multidimensional
  - array object
- To import numpy and call numpy, in convention:
  - import numpy as np

Array Creation by Initializer

- n = np.array([1, 2, 3, 4])
- n = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
- The number of dimension is called rank
- The number of the size of each dimension is called shape.

Array Creation by Functions

- Create dimension a * b and fill with 0:
  - n = np.zeros((a, b))
- Create dimension a * b and fill with 1:
  - n = np.ones((a, b))
- Create dimension a * b and fill with x:
  - n = np.full((a, b), x)
- Create dimension a * b and fill with a random float in [0, 1]:
  - n = np.random.random((a, b))
- Create dimension a * b and fill with 0 to a*b -1:
  - n = np.arange(a*b-1).reshape(a, b)

Let's Do

- At Google Colab

Array Indexing

- We can use similar slicing methods in Python lists.
  - However, the slice is just a view (reference) to the original, not a new one.
- For a n*m matrix, we can get a sub matrix by slicing rows and columns.
- e.g., if a = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]),
  - r1 = a[1, :] will give you one row of the matrix - rank as 1
  - r2 = a[1:2, :] will give you one matrix with one row - rank as 2
  - c1 = a[:, 1] will give you one column of the matrix - rank as 1
  - c2 = a[:, 1:2] will give you one matrix with one column - rank as 2
  - m1 = a[1, 1] will give one cell - rank as 0
  - m2 = a[1:2, 1:2] will give you one matrix with one row/column - rank as 2

Let's Do

- At Google Colab

Array Math

- Arithmetic operations are element wise:
  - +: a + b, or np.add(a, b)
  - -: a - b, or np.substract(a, b)
  - *: a * b, or np.multiply(a, b)
  - /: a / b, or np.divide(x, y)
  - sqrt: np.sqrt(a)
  - Absolute value: np.absolute(a)
  - Matrix multiplication: a.dot(b) or np.dot(a, b), or a@b

Array Math

- Some other useful functions
  - sum of all: a.sum() or np.sum(a)
  - sum of columns: a.sum(axis = 0) or np.sum(a, axis = 0)
  - sum of rows: a.sum(axis = 1) or np.sum(a, axis = 1)
  - Transpose of matrix: a.T
  - minimun: a.min(), a.min(axis = 0), a.min(axis = 1)

Advanced Indexing

- a = np.arrange(10)**3
- i = [0, 1, 2, 4, 8]
- a[i] will loop a for index i in [0, 1, 2, 4, 8]
- b = a > 20 will do elementweise comparison and return a shape of a with boolean.
- a[b] will print elements where b has True.
- np.where(condition) will return the index where condition is True

Exercise

- Create an array with 100 random floats
- Print the elements > 0.5
- Print the index of elements > 0.5
- Replace the elements < 0.5 to be -1

Exercise

- Create an array with 100 random integers
  - a=np.random.randint(0, 100,(100))
- Normalize the array to [0, 1] using min/max method:
  - hint: value_new = (value_old - min)/(max-min)

Exercise

- Create an array with 100 random integers
- Find the percentile of the array Q1 = 25, Q2 = 50, Q3 = 75

More Practice

- https://www.machinelearningplus.com/python/101-numpy-exercises-python/

Thank you!