In-class assignments for Section 1: numpy

Subhasis Ray[*]

<2022-09-21 Wed>

Save your work in a file named `W04P1_classwork.py` and submit it to codePost.

The first deadline for submitting your work on codePost is 1:30 PM. Do this to get attendance. Don't worry about passing the tests.

The submission will open again in the afternoon, and you have until midnight to submit an updated version. Do not hesitate to resubmit after you have solved each part. That way, you get feedback from the tests, and even if you cannot finish all of them, it is better than missing the deadline by a few minutes.

1 Startup

Import numpy.

```
import numpy as np
```

In this assignment, you should not use any python loops. They are slow. The main purpose of numpy is to provide vectorized operations which are much faster. Utilize numpy functions and operations wherever possible.

2 Array creation and indexing: 2 points

There are several ways to create arrays in `numpy`. Check the functions `array`, `zeros`, `ones`, `empty`, `arange` and `linspace`. Also, take a look at `random` submodule to find out how to create arrays filled with random numbers. See the user guide here: `https://numpy.org/doc/stable/user/basics.creation.html`

[*]subhasis.ray@plaksha.edu.in

See the user guide on indexing ndarrays: `https://numpy.org/doc/stable/user/basics.indexing.html`

Now create an array `array2d` which has the numbers 0-19 (inclusive) sequentially in 4 rows and 5 columns (see printed output below). You can use `arange` followed by `reshape`.

3 Creating a random array and replacing certain rows and columns: 3 points

Now, create a (pseudo) random number generator (RNG) using the following code:

```
rng = np.random.default_rng(0)
print(rng.random())
```

The argument 0 to `default_rng` *seeds* the RNG. This ensures that the same sequence of random numbers are generated every time. You should see the same output as above. Remember, each time you ask an RNG to generate a random number, it moves to the next random number in the sequence. To get back to the same spot, you will have to reseed it.

Now, create 5x4 array filled with random uniform floats. Note that you have to put the dimensions of the array as a `tuple` argument to `rng.random()`.

```
rand_array = rng.random((5, 4))
```

Now replace (counting from 1) the 2nd row in `array2d` with the 3rd column of `rand_array`. Also add to the 1st column of `array2d` the 4th row of `rand_array`.

This should raise an exception. Do you see why?

Now comment out the problematic code, and convert `array2d` into a floating point array. You may use `astype` function.

4 Stats on arrays: 5 points

Now create these variables from `array2d`:

- `value_range` a 2-tuple containing the minimum and the maximum value
- `std` the standard deviation across all entries (go with the default, dividing the sum-of-square errors by n, not n-1)

- `mean_rows` a 1D array containing mean of each row
- `sum_cols` a 1D array containing the sum of each column
- `matrix_product` the matrix product `array2d` x `rand_array`.

Some of these functions take a keyword argument `axis` which might be helpful.

5 Transpose

Now create another variable `array2d_T` containing the transpose of `array2d`.

```
array2d_T = array2d.transpose()
print('transpose:\n', array2d_T)
```

```
transpose:
 [[ 0.03358558  0.72965545 10.17565562 15.86317892]
 [ 1.          0.         11.         16.        ]
 [ 2.          0.         12.         17.        ]
 [ 3.          0.         13.         18.        ]
 [ 4.          0.         14.         19.        ]]
```

6 Concatenation: 2 points

Look up the documentation for `np.concatenate`. Now concatenate this transpose `array2d_T` with `rand_array` to create a new array `concated` of shape `(5, 8)`. What happens if you try to concatenate `array2d` with `rand_array`?

7 Stacking: 2 points

There is another way of putting together two arrays. Think of a 2D array as a slab. You can stack them one on top of the other to make a 3D array. Look up the docs for `stack` function. Now stack `array2d_T` and `rand_array` in that order. Notice the `axis` keyword argument? Specify it so that the new dimension is the first dimension. Store the result in the variable `stacked`.

8 Another way of stacking: 1 points

Now specify the `axis` argument so that the new dimension is the last one. Store the result in a variable named `stacked2`.

9 Flip the array elements: 2 points

Read the docs on `flip` function. Now create another array `flipped` reversing the order of elements in each row of `rand_array`.

10 Structured array: 10 points

Look up the user guide on numpy structured arrays or record arrays: `https://numpy.org/doc/stable/user/basics.rec.html#`. Pay attention to how new structured data types are specified using `dtype`. Below are three lists, `names`: the names of cities, `lats`: their latitudes, and `longs`: their longitudes. Create a structured array called `city_loc` with three fields, `name`: a string containing the name of the city, `lat`: a number storing the latitude, and `long`: number storing the longitude (notice that no more plurals). Make the `name` field 16 unicode chars long (in the `dtype`, specify `'U16'` for this field).

```
names = [
    'Tokyo',
    'Jakarta',
    'Delhi',
    'Manila',
    'Sao Paulo',
    'Seoul',
    'Mumbai',
    'Shanghai',
    'Mexico City',
    'Guangzhou',
    'Cairo',
    'Beijing',
    'New York',
    'Kolkata',
    'Moscow',
    'Bangkok',
    'Dhaka',
    'Buenos Aires'
]

lats = [
    35.6839,
    -6.2146,
    28.6667,
    14.6,
    -23.5504,
    37.56,
    19.0758,
    31.1667,
    19.4333,
    23.1288,
    30.0444,
    39.904,
    40.6943,
    22.5727,
    55.7558,
    13.75,
    23.7289,
    -34.5997
]

longs = [
    139.7744,
    106.8451,
    77.2167,
    120.9833,
    -46.6339,
    126.99,
    72.8775,
    121.4667,
    -99.1333,
    113.259,
    31.2358,
    116.4075,
    -73.9249,
    88.3639,
    37.6178,
    100.5167,
    90.3944,
    -58.3819
]
```

11 Query the structured array: 1 + 2 = 3 points

11.1 Southern hemisphere cities: 1 point

Now create a variable `southern` with the names of the cities in the southern hemisphere.

11.2 Cities in southern and western hemisphere: 2 points

For this you will need an 'AND' operation. Notice that instead of the Python `and`, here you need `&` operator. Store the result in a variable named `sowthwest`

12 Submit your script to codePost.

If you are using jupyter notebook, go to File menu, download as python (.py) file and submit that. The filename should be `W04P1_classwork.py`.

13 Other useful functions:

You may want to look up the docs for the following:

- `flatten()`
- `meshgrid()`
- `r_[]`
- `c_[]`
- `ravel()`
- `nonzero()`
- `where()`

- `allclose()`
- `isnan()`
- `char.*` set of functions
- The functions / operators mentioned here: `https://numpy.org/doc/stable/user/numpy-for-matlab-users.html`