hw01_teng_gradescope

September 17, 2019

0.1 Homework 1

Please import the following packages.

[2]:
```
import numpy as np
import scipy.sparse
```

Please download `memory.py` from Resources/Homework/Homework01 on NYU Classes. Save it to the same directory as the Jupyter notebook. Please import the following package.

[3]:
```
import memory
```

0.1.1 Loops

How to go through the entries of an array top to bottom/left to right?

1. Given an array `inputArray`, write a `for` loop that flattens it to `outputArray`. For example, `inputArray = np.array([[1,2], [3,4]])` would yield `np.array([1,2,3,4])` for `outputArray`.

[4]:
```
inputArray = np.array([[1,2],[3,4]])

outputArray = []
for row in inputArray:
    for elem in row:
        outputArray.append(elem)

outputArray = np.array(outputArray)

print(outputArray)
print(type(outputArray))
```

```
[1 2 3 4]
<class 'numpy.ndarray'>
```

2. Given a jagged array `inputArray`, write a `for` loop that flattens it to `outputArray`. For example, `inputArray = np.array([[1,2,3], [4]])` would yield `np.array([1,2,3,4])` for `outputArray`.

[6]:
```
inputArray = np.array([[1,2,3], [4]])

outputArray = []
for i in inputArray:
    for j in i:
        outputArray.append(j)
outputArray = np.array(outputArray)


print(outputArray)
print(type(outputArray))
```

```
[1 2 3 4]
<class 'numpy.ndarray'>
```

0.1.2 Packages

How to import and use packages?

3. Create an array `A` from the list

   `[[1, 0, 0, 1, 0, 0], [0, 0, 2, 0, 0, 1], [0, 0, 0, 2, 0, 0]]`

   Use `memory.getsizeof` to determine how much space `A` takes up in memory.

[7]:
```
A = [[1, 0, 0, 1, 0, 0], [0, 0, 2, 0, 0, 1], [0, 0, 0, 2, 0, 0]]
print(A)
print(type(A))

A = np.asarray(A)
print(A)
print(type(A))

memory.getsizeof(A)
```

```
[[1, 0, 0, 1, 0, 0], [0, 0, 2, 0, 0, 1], [0, 0, 0, 2, 0, 0]]
<class 'list'>
[[1 0 0 1 0 0]
 [0 0 2 0 0 1]
 [0 0 0 2 0 0]]
<class 'numpy.ndarray'>
```

[7]:
```
144
```

4. Use `scipy.sparse.csr_matrix` to covert `A` into `S`. Use `memory.getsizeof` to determine how much space `S` takes up in memory.

[8]:
```
S = scipy.sparse.csr_matrix(A)
print(S)

memory.getsizeof(S)
```

```
  (0, 0)        1
  (0, 3)        1
  (1, 2)        2
  (1, 5)        1
  (2, 3)        2
```

[8]:
```
76
```

5. What accounts for the difference? Try calling `print` on `S`.

[9]:
```
print(S)
```

```
  (0, 0)        1
  (0, 3)        1
  (1, 2)        2
  (1, 5)        1
  (2, 3)        2
```

[10]:
```
# The difference in storage is caused by the space saved when using a sparse␣
 ↪matrix vs a dense matrix. (https://machinelearningmastery.com/
 ↪sparse-matrices-for-machine-learning/).
# Sparse matrices are comprised mostly of 0 values. We are not saving the 0's␣
 ↪in a sparse matrix.
```

[ ]:
```

```