assignment-1-questions

February 23, 2023

0.1 Assignment 1

Submit this notebook in the Google classroom by replacing the filename with your Roll number followed by name.

We have extensively learned *Numpy* in the class. In this assignment, our objective is to appreciate the time gain of using vectorisation operation using Numpy and get a hands on experience with *Numpy* broadcasting

Question 1 - Perform matrix multiplication using vanilla Python. Fill in the blank lines of code - Use `timeit` to find out the time taken to multiply the matrices

[34]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[27]:
```
# vanilla python code to multiply two matrix
def matmul_python(x, y):
    '''
    Function to multiply two matrix without using vectorisation
    Input Arguments:
        x: a list of lists that reprepents a 2d matrix
        y: a list of lists that represents a 2d matrix
    Returns:
        a resultant list of list
    '''
    result = [[0]*y.shape[1]]*x.shape[0] #result list to store the result of␣
 ↪multiplication
    colY = # write your code here
    rowX = # write your code here
    for row in range(rowX):
        for col in #write your code here:
            for i,itemX in enumerate(x[row]):
                result[row][col]+= # write your code here
    return result

# create the matrix to multiply
np.random.seed(0) #do not change this code
a = np.random.randint(2,5,(600,412))
b = np.random.randint(0,2,(412,741))
```

```
%timeit matmul_python(a,b)
```

```
1min 4s ± 115 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

- Now execute a single line of code in numpy to perform matrix multiplication
- Use `timeit` to find out the time taken to multiply the matrices.

```
[29]: np.random.seed(0) #do not change this code
      a = np.random.randint(2,5,(600,412))
      b = np.random.randint(0,2,(412,741))
      %timeit np.matmul(a,b)
```

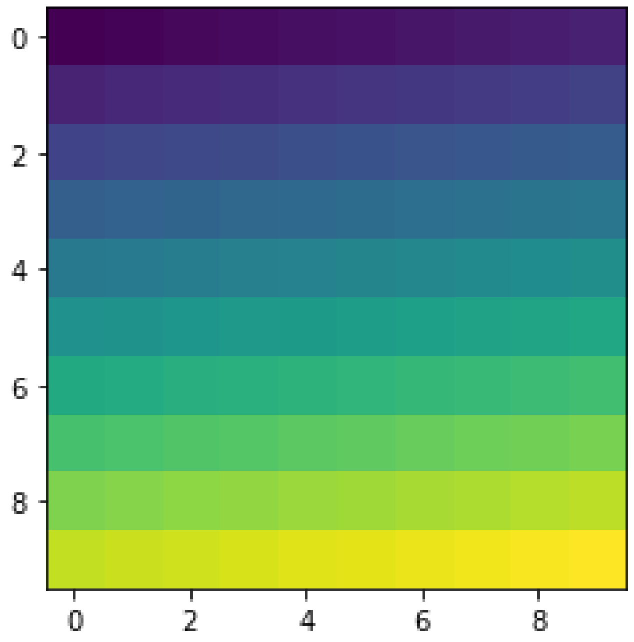
```
128 ms ± 18.8 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

```
[ ]: difference = #write your code here. The time difference between both the␣
       ↪execution
     print("Python method of multplying matrix and numpy method has a difference of␣
       ↪{} seconds".format(difference))
```

Question 2 - Use *Numpy* and *plt.imshow()* to create a 2D matrix that plots the following figure . - Hint: This is a 10x10 matrix with contigiously increasing values from 0 to 99. The value at first row and first column is 0 and the value at last

```
[44]: np.random.seed(0) #do not change this code
      matrix = # write your code here
      plt.imshow(matrix)
```

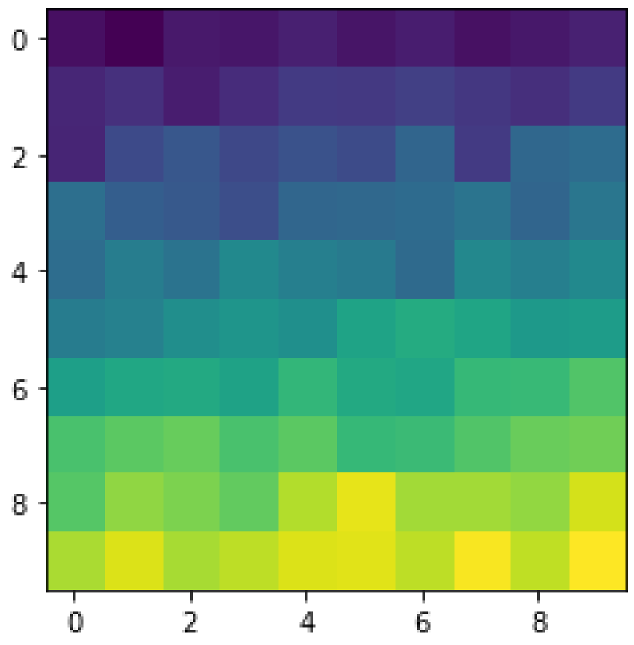
```
[44]: <matplotlib.image.AxesImage at 0x7fbe73588250>
```



- Now add noise to each element of the *matrix* variable. The noise should be from a normal distribution with mean of 5.5 and standard deviation of 3. Draw the matrix variable again.

[46]:
```
np.random.seed(0)
noise = # write your code here
matrix = matrix + noise
plt.imshow(matrix)
```

[46]:
```
<matplotlib.image.AxesImage at 0x7fbe60ef3370>
```



Question 3 - In this problem, we require need to create a 2d matrix (n x m) with random floating point numbers. Here m = 180. - We need to ensure that the first 120 elements are in range 20.0-30.0 and remaining elements are in the range 80-90. Let n=50

[62]:
```
# first create the 50x120 matrix for the first 120 elements.
# create this matrix from uniform distribution for the given range
np.random.seed(0) # do not remove this line of code
p1 = # write your code here
print(p1.shape)

np.random.seed(0) # do not remove this line of code
#now create another matrix of size 50x60 for the reamining element that span␣
  ↪80-90
p2 = # write your code here
```

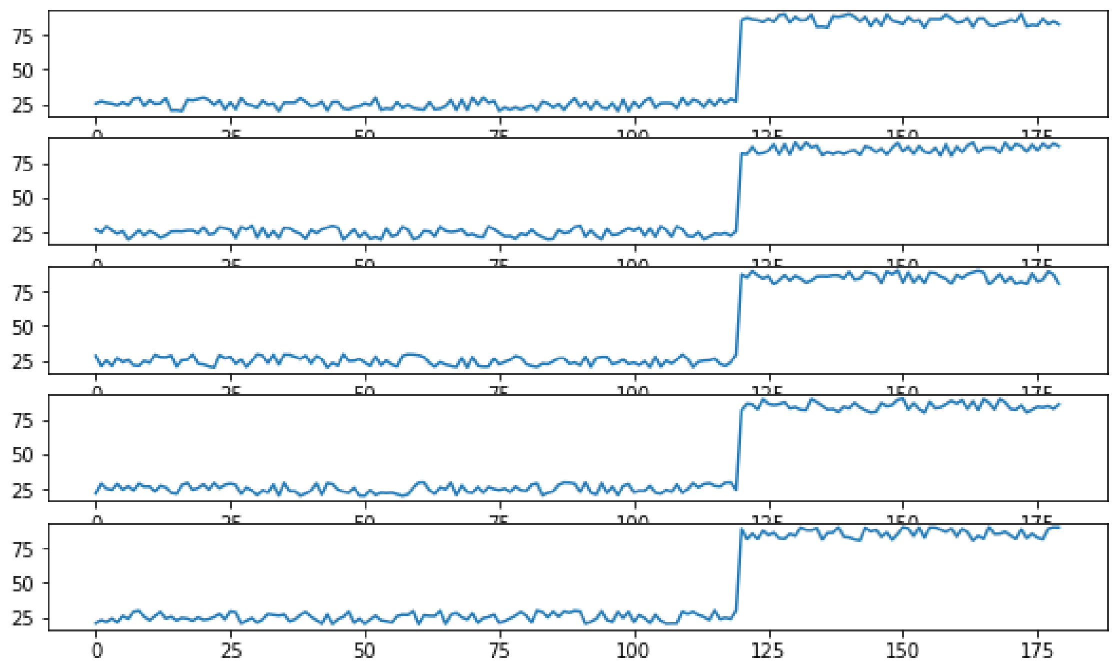
```
print(p2.shape)

# now use broadcasting to use p1 (50, 120) and p2 (50,60) to convert it to␣
  ↪(50,180)
p_final = # write your code here

print(p_final.shape)
```

```
(50, 120)
(50, 60)
(50, 180)
```

[68]:
```
# plot the first 5 rows of _p_final_ in a single plot. The choise of plot␣
  ↪should be a line plot
fig, ax = plt.subplots(5,1, figsize=(10,6))
for i in range(5):
    # write your code here
```



[ ]:
```
```