Lab3

September 17, 2024

1 Lab 3

Labs in general are for you to solve short programming challenges in class. In contrast, homework assignments will involve more challenging and lengthy problems.

Feel free to ask the TAs for help if there is anything you do not understand. The TAs will go through suggested solutions in the last 15 minutes of the lab - typically by solving them in a live demo.

Please remember to submit the lab on Gradescope.

1.0.1 A few points to review

1. NumPy

[ ]:
```
import numpy as np
```

1. (1 point). Create two lists of 100 random integers in the `range[0, 1000]`. Find all numbers that are shared by the two lists.

[ ]:
```
list1 = np.random.randint(0, 1000, size=100)
list2 = np.random.randint(0, 1000, size=100)
```

[ ]:
```
list3 = []
for n in list1:
    if n in list2:
        list3.append(n)
print(list3)
```

```
[208, 371, 973, 391, 334, 575, 666, 66, 824, 753, 990]
```

2. (1.5 point). Given a NumPy array `data = np.random.randint(0, 100, size=(10, 10))`, perform the following tasks:

- Create a boolean mask to filter out values greater than 50.
- Use this mask to extract the filtered values.
- Compute the mean of the filtered values.

[ ]:
```
data = np.random.randint(0, 100, size=(10,10))
mask = data > 50
filtered_data = data[mask]
```

```
np.mean(filtered_data)
```

```
[ ]: 78.56818181818181
```

3. (3 points) Basic array manipulation.

- Generate a random matrix with shape (4,5) with draws from a standard normal distribution
- Find row sums
- Scale so each row has min=0 and max=1 (for a challenge, do this in one line of code)
- Find row sums when you exclude the max in each row

```
[ ]: matrix = np.random.normal(size=(4,5))
     matrix
```

```
[ ]: array([[ 0.01189355, -2.4714661 ,  0.72633183,  0.126992  ,  0.50263136],
            [-0.92192167,  0.62049476,  0.30306809,  0.27118836, -0.19852311],
            [-1.05555982, -1.1171553 , -0.18271588,  0.08048342,  0.84417538],
            [ 2.23740111, -0.2988071 ,  0.71375248, -0.48570843,  0.41014135]])
```

```
[ ]: row_sums = matrix.sum(axis=1)
     row_sums
```

```
[ ]: array([-1.10361736,  0.07430642, -1.43077221,  2.5767794 ])
```

```
[ ]: scaled_matrix = (matrix - matrix.min(axis=1, keepdims=True)) / (matrix.
      ↪max(axis=1, keepdims=True) - matrix.min(axis=1, keepdims=True))
     scaled_matrix
```

```
[ ]: array([[0.77658429, 0.        , 1.        , 0.81257733, 0.93004546],
            [0.        , 1.        , 0.79420171, 0.77353302, 0.46900341],
            [0.03140494, 0.        , 0.47643135, 0.6106256 , 1.        ],
            [1.        , 0.06863526, 0.44047472, 0.        , 0.32898044]])
```

```
[ ]: row_sums - matrix.max(axis=1)
```

```
[ ]: array([-1.8299492 , -0.54618833, -2.27494759,  0.33937829])
```

```
[ ]:
```

```
[ ]:
```

4. (2 points) Create two NumPy arrays: a 2D array `matrix = np.arange(1, 13).reshape(3, 4)` and a 1D array `vector = np.array([1, 2, 3, 4])`. Perform the following operations using broadcasting:

- Add the vector to each row of the matrix.
- Subtract the vector from each column of the matrix.
- Multiply each element of the matrix by the vector element-wise.
- Return the results of each operation.

[ ]:
```
matrix = np.arange(1, 13). reshape(3,4)
vector = np.array([1,2,3,4])
matrix
```

[ ]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

[ ]:
```
add = matrix + vector
add
```

[ ]:
```
array([[ 2,  4,  6,  8],
       [ 6,  8, 10, 12],
       [10, 12, 14, 16]])
```

[ ]:
```
sub = matrix - vector
sub
```

[ ]:
```
array([[0, 0, 0, 0],
       [4, 4, 4, 4],
       [8, 8, 8, 8]])
```

[ ]:
```
mult = matrix * vector
mult
```

[ ]:
```
array([[ 1,  4,  9, 16],
       [ 5, 12, 21, 32],
       [ 9, 20, 33, 48]])
```

5. (0.5 points). You have two datasets `x = np.array([1, 2, 3, 4, 5])` and `y = np.array([2, 4, 6, 8, 10])`. Compute the Pearson correlation coefficient between `x` and `y`.

Hint: use `np.corrcoef()`

[ ]:
```
x = np.array([1, 2, 3, 4, 5])
y = np.array([2, 4, 6, 8, 10])
np.corrcoef(x,y)[1,0] # pulling out x,y from correlation matrix
```

[ ]:
```
0.9999999999999999
```

6 (2 point). You are working with a dataset of daily sales figures for a retail store over a year. The dataset is represented as a 1D NumPy array `sales_data = [200, 220, 250, 275, 300, 320, 310, 290, 270, 260]`, where each element corresponds to the sales for one day.

- Compute the cumulative sum of the sales data and print the total sales accumulated up to each day.
- Determine the 25th, 50th, and 75th percentiles (quartiles) of the daily sales data.

```
[ ]: sales_data = np.array([200, 220, 250, 275, 300, 320, 310, 290, 270, 260])
     cum_sales = sales_data.cumsum()
     quartiles = np.percentile(sales_data, [25, 50, 75])
```

```
[ ]: cum_sales
```

```
[ ]: array([ 200,  420,  670,  945, 1245, 1565, 1875, 2165, 2435, 2695])
```

```
[ ]: quartiles
```

```
[ ]: array([252.5, 272.5, 297.5])
```