Assignment - 2 (ML)

Nishant Parekh - J046

Task1: Prove multiplication properties of matrices

In [13]:

```
import numpy as np
P= np.array([[5,1,2],[7,8,6],[75,5,69]])
Q= np.array([[3,8,12],[11,12,14],[8,9,11]])
R= np.array([[3,1,11],[45,46,42],[58,61,60]])
I= np.identity(3)
```

In [14]:

```
print('Matrix P: \n',P)
print('Matrix Q: \n',Q)
print('Matrix R: \n',R)
print('Identity Martix: \n',I)
```

```
Matrix P:
 [[ 5  1  2]
 [ 7  8  6]
 [75  5 69]]
Matrix Q:
 [[ 3  8 12]
 [11 12 14]
 [ 8  9 11]]
Matrix R:
 [[ 3  1 11]
 [45 46 42]
 [58 61 60]]
Identity Martix:
 [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

Commutative Property (does not hold true):

In [15]:

```
P_dot_Q = P.dot(Q)
Q_dot_P = Q.dot(P)
```

In [16]:

```
print('P.Q: \n',P_dot_Q)
print('Q.P: \n',Q_dot_P)
```

```
P.Q:
 [[  42   70   96]
 [ 157  206  262]
 [ 832 1281 1729]]
Q.P:
 [[ 971  127  882]
 [1189  177 1060]
 [ 928  135  829]]
```

Associative property:

In [17]:

```
PQ_R = np.dot(P, Q).dot(R)
P_QR = P.dot(np.dot(Q, R))
```

In [18]:

```
print('(P.Q).R: \n',PQ_R)
print('(P.(Q.R): \n)',P_QR)
```

```
(P.Q).R:
 [[  8844   9118   9162]
 [ 24937  25615  26099]
 [160423 165227 166694]]
(P.(Q.R):
) [[  8844   9118   9162]
 [ 24937  25615  26099]
 [160423 165227 166694]]
```

Distributive property:

In [19]:

```
lhs = np.dot(P, Q+R)
rhs = np.dot(P, Q) + np.dot(P, R)
```

In [20]:

```
print('P.(Q+R): \n',lhs)
print('P.Q + P.R: \n',rhs)
```

```
P.(Q+R):
 [[ 218  243  313]
 [ 886  947 1035]
 [5284 5795 6904]]
P.Q + P.R:
 [[ 218  243  313]
 [ 886  947 1035]
 [5284 5795 6904]]
```

Identity property:

In [21]:

```
PI = np.dot(P, I)
IP = np.dot(I, P)
```

In [22]:

```
print('P.I: \n',PI)
print('I.P: \n',IP)
```

```
P.I:
 [[ 5.  1.  2.]
 [ 7.  8.  6.]
 [75.  5. 69.]]
I.P:
 [[ 5.  1.  2.]
 [ 7.  8.  6.]
 [75.  5. 69.]]
```

Multiplicative property of zeros:

In [23]:

```
a = np.zeros(9).reshape(3, 3)
lhs = np.dot(P, a)
rhs = np.dot(a, P)
```

In [24]:

```
print('P.0: \n', lhs)
print('0.P: \n',rhs)
```

```
P.0:
 [[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
0.P:
 [[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

Dimensions on matrix multiplication:

In [32]:

```
a,b,c = 2,4,6
mat_a_b = np.random.randn(a, b)
mat_b_c = np.random.randn(b, c)
mat_multi = np.dot(mat_a_b, mat_b_c)
result_x, result_y = mat_multi.shape
```

In [33]:

```
print(f'{a}x{b} matrix X {b}x{c} matrix = {result_x}x{result_y} matrix')
```

```
2x4 matrix X 4x6 matrix = 2x6 matrix
```

Task2: Inverse of a matrix:

In [34]:

```
Q_inv = np.linalg.inv(Q)
Q_inv
```

Out[34]:

```
array([[-0.33333333, -1.11111111,  1.77777778],
       [ 0.5       ,  3.5       , -5.        ],
       [-0.16666667, -2.05555556,  2.88888889]])
```

Task3: Comparison of time between numpy and loops:

In [35]:

```
import time
size = 3000
numpy_mat_A = np.random.randn(size, size)
numpy_mat_B = np.random.randn(size, size)
list_mat_A = [list(i) for i in numpy_mat_A]
list_mat_B = [list(i) for i in numpy_mat_B]
```

In [36]:

```
loop_st = time.time()
list_mat_C = []
for i in range(size):
    row = []
    for j in range(size):
        row.append(list_mat_A[i][j] + list_mat_B[i][j])
    list_mat_C.append(row)
loop_ed = time.time()
```

In [37]:

```
numpy_st = time.time()
```

```
numpy_mat_C = numpy_mat_A + numpy_mat_B
numpy_ed = time.time()
```

In [38]:

```
print('Loops: ', loop_ed - loop_st)
print('Numpy: ', numpy_ed - numpy_st)
```

```
Loops:  5.0635058879852295
Numpy:  0.1387498378753662
```

Conclusion: Numpy is way much faster than the normal loops.

In [ ]:

```

```