Assignment 2

Name: Ishan Jhanwar

Roll: J024

Date: 24/7/2021

Task 1: Prove properties of matrix multiplication

In [1]:
```
import numpy as np
np.random.seed(1729)
A = np.random.randn(3, 3)
B = np.random.randn(3, 3)
C = np.random.randn(3, 3)
I = np.identity(3)
```

In [2]:
```
print('Matrix A:\n', A, end='\n\n')
print('Matrix B:\n', B, end='\n\n')
print('Matrix C:\n', C, end='\n\n')
print('Identity Matrix:\n', I)
```

```
Matrix A:
 [[-0.68733944 -0.82099471  1.65236086]
 [-0.57529304 1.09896774  0.92594603]
 [-0.99341379 -0.85822114  0.07488676]]

Matrix B:
 [[ 0.52935554 0.12095155 -0.22442361]
 [-1.55667849 0.05594088  0.16147154]
 [-2.13464176 0.10967005  0.44301216]]

Matrix C:
 [[ 0.39626623 0.24979741  1.29849737]
 [-1.28043367 -0.97546584 -0.26908664]
 [-1.10573836 -0.12799271 -0.61782738]]

Identity Matrix:
 [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

Commutative Property (does not hold true, i.e. A.B != B.A)

In [3]:
```
A_dot_B = A.dot(B)
B_dot_A = B.dot(A)

print('A.B:\n', A_dot_B, end='\n\n')
print('B.A:\n', B_dot_A)
```

```
A.B:
 [[-2.61302062 0.05215256  0.75370387]
 [-3.99183705 0.09344318 0.7167667 ]
 [ 0.65024889 -0.15995175  0.11754297]]

B.A:
 [[-0.21048402 -0.10907116  0.96987463]
 [ 0.87737608 1.20092375 -2.5083043 ]
 [ 0.96403667 1.49285104 -3.3924742 ]]
```

Clearly, A.B != B.A

Associative Property - (A.B)C = A.(B.C)

In [4]:
```
AB_C = np.dot(A, B).dot(C)
A_BC = A.dot(np.dot(B, C))

print('(A.B)C\n', AB_C, end='\n\n')
print('A(B.C)\n', A_BC)
```

```
(A.B)C
 [[-1.935629   -0.80006741 -3.87269286]
 [-2.49403444 -1.18004209 -5.65137233]
 [ 0.33250751 0.30341331  0.81476609]]

A(B.C)
 [[-1.935629   -0.80006741 -3.87269286]
 [-2.49403444 -1.18004209 -5.65137233]
 [ 0.33250751 0.30341331  0.81476609]]
```

As both result matrix are equal, Associative Property holds true for matrices.

Distributive Property A(B + C) = AB + AC

In [5]:
```
A_B_C = np.dot(A, B + C)
AB_AC = np.dot(A, B) + np.dot(A, C)

print('A(B + C)\n', A_B_C, end='\n\n')
print('AB + AC\n', AB_AC)
```

```
A(B + C)
 [[-3.66123955 0.4698191  -0.93875967]
 [-6.65081559 -1.24078335 -0.89804214]
 [ 1.27268263 0.41947651 -0.98773348]]

AB + AC
 [[-3.66123955 0.4698191  -0.93875967]
 [-6.65081559 -1.24078335 -0.89804214]
 [ 1.27268263 0.41947651 -0.98773348]]
```

As both result matrix are equal, Distributive Property holds true for matrices.

Multiplicative identity property IA = AI

In [6]:
```
IA = np.dot(I, A)
AI = np.dot(A, I)

print('I.A\n', IA, end='\n\n')
print('A.I\n', AI)
```

```
I.A
 [[-0.68733944 -0.82099471  1.65236086]
 [-0.57529304 1.09896774  0.92594603]
 [-0.99341379 -0.85822114  0.07488676]]

A.I
 [[-0.68733944 -0.82099471  1.65236086]
 [-0.57529304 1.09896774  0.92594603]
 [-0.99341379 -0.85822114  0.07488676]]
```

As both result matrix are equal, Multiplicative identity property holds true for matrices.

Multiplicative property of zero - A.0 = 0.A = 0

In [7]:
```
zero_mat = np.zeros(9).reshape(3, 3)

dot_0_A = np.dot(zero_mat, A)
dot_A_0 = np.dot(A, zero_mat)

print('A.0 is equal to 0:', np.array_equal(zero_mat, dot_0_A))
print('0.A is equal to 0:', np.array_equal(zero_mat, dot_A_0))
```

```
A.0 is equal to 0: True
0.A is equal to 0: True
```

Hence, the property is proved.

Dimension property - The product of an m×n matrix and nxk matrix is an m×k matrix.

In [8]:
```
m, n, k = 5, 7, 3

mat_m_n = np.random.randn(m, n)
mat_n_k = np.random.randn(n, k)

mat_mult = np.dot(mat_m_n, mat_n_k)
result_x, result_y = mat_mult.shape

print(f'The product of a {m}x{n} matrix and {n}x{k} matrix gave a {result_x}x{result_y} matrix')
```

```
The product of a 5x7 matrix and 7x3 matrix gave a 5x3 matrix
```

Hence the property is proved.

Task 2 - Matrix Inverse

In [9]:
```
A_inv = np.linalg.inv(A)
A_inv
```

Out[9]:
```
array([[ 0.32043465, -0.49569237, -0.94127846],
       [-0.32036199,  0.5809731 , -0.11478837],
       [ 0.57931164, 0.08246802, -0.44858147]])
```

Task 3 - Numpy vs Loops

In [10]:
```
import time

size = 5000
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 [11]:
```
start_loop = 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)
end_loop = time.time()
```

In [12]:
```
start_numpy = time.time()
numpy_mat_C = numpy_mat_A + numpy_mat_B
end_numpy = time.time()
```

In [13]:
```
print(f'Loops took {end_loop - start_loop} seconds while Numpy took {end_numpy - start_numpy} seco
nds')
```

```
Loops took 7.678518533706665 seconds while Numpy took 0.7266678810119629 seconds
```

This shows how ridiculously fast Numpy computations are.