ML Assignment 2

1:Prove properties of matrix multiplication

In [1]:

```
import numpy as np

a = np.array([[1,2,3],[4,5,6],[7,8,9]])
b = np.array([[1,4,1],[8,2,3],[4,1,9]])
c = np.array([[3,1,2],[3,5,2],[1,1,1]])
I = np.identity(3)
```

In [2]:

```
print('Matrix A : \n', a)
print('Matrix B : \n', b)
print('Matrix C : \n', c)
print('Identity Matrix : \n', I)
```

```
Matrix A :
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
Matrix B :
 [[1 4 1]
 [8 2 3]
 [4 1 9]]
Matrix C :
 [[3 1 2]
 [3 5 2]
 [1 1 1]]
Identity Matrix :
 [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

Communatitive property not applicable

In [3]:

```
AdotB = a.dot(b)
BdotA = b.dot(a)

print('A.B : \n', AdotB)
print('B.A : \n', BdotA)
```

```
A.B :
 [[ 29  11  34]
 [ 68  32  73]
 [107  53 112]]
B.A :
 [[24 30 36]
 [37 50 63]
 [71 85 99]]
```

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)
print('A.(B.C) : \n', A_BC)
```

```
(A.B).C :
 [[154 118 114]
 [373 301 273]
 [592 484 432]]
A.(B.C) :
 [[154 118 114]
 [373 301 273]
 [592 484 432]]
```

Distributive property [A.(B+C) =]

In [5]:

```
lhs = np.dot(a,b + c)
rhs = np.dot(a, b) + np.dot(a, c)
print("A.(B+C) : \n", lhs)
print('A.B + A.C : \n', rhs)
```

```
A.(B+C) :
 [[ 41  25  43]
 [101  67  97]
 [161 109 151]]
A.B + A.C :
 [[ 41  25  43]
 [101  67  97]
 [161 109 151]]
```

Identity property [A.I = I.A]

In [6]:

```
AI = np.dot(a, I)
IA = np.dot(I, a)

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

```
A.I :
 [[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]
I.A :
 [[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]
```

Multiplicative property of zero [A.0 = 0.A = 0]

In [7]:

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

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

```
A.0 :
 [[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
0.A :
 [[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

Dimensions on matrix multiplication

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' {m}x{n} matrix X {n}x{k} matrix = {result_x}x{result_y} matrix')
```

```
 5x7 matrix X 7x3 matrix = 5x3 matrix
```

2: Inverse of a matrix

In [9]:

```
A_inv=np.linalg.inv(a)
A_inv
```

Out[9]:

```
array([[ 3.15251974e+15, -6.30503948e+15,  3.15251974e+15],
       [-6.30503948e+15,  1.26100790e+16, -6.30503948e+15],
       [ 3.15251974e+15, -6.30503948e+15,  3.15251974e+15]])
```

3:Comparison of time between numpy and 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('Time for loops : ', end_loop - start_loop)
print('Time for numpy : ', end_numpy - start_numpy)
```

```
Time for loops :  20.560731887817383
Time for numpy :  4.3703203201293945
```

In [1]:

```
## Numpy is faster than loops
```

In [ ]:

```

```