Machine Learning Assignment

Aditya Gavankar (J072)

Exp 2 :- Numpy Basics & Properties

In [1]:

```
import numpy as np
import time
```

In [2]:

```
mat1 = np.arange(11, 20, 1)
A = mat1.reshape(3, 3)
A
```

Out[2]:

```
array([[11, 12, 13],
       [14, 15, 16],
       [17, 18, 19]])
```

In [3]:

```
mat2 = np.arange(10, 19, 1)
B = mat2.reshape(3, 3)
B
```

Out[3]:

```
array([[10, 11, 12],
       [13, 14, 15],
       [16, 17, 18]])
```

In [4]:

```
mat3 = np.arange(0, 9, 1)
C = mat3.reshape(3, 3)
C
```

Out[4]:

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Task 1 :- Properties

1) Non-commutitive (AB != BA)

In [5]:

```
np.dot(A, B)
```

Out[5]:

```
array([[474, 510, 546],
```

```
       [591, 636, 681],
       [708, 762, 816]])
```

In [6]:

```
np.dot(B, A)
```

Out[6]:

```
array([[468, 501, 534],
       [594, 636, 678],
       [720, 771, 822]])
```

2) Associative [A(BC) = (AB)C]

In [7]:

```
A.dot(B.dot(C))
```

Out[7]:

```
array([[ 4806,  6336,  7866],
       [ 5994,  7902,  9810],
       [ 7182,  9468, 11754]])
```

In [8]:

```
(A.dot(B)).dot(C)
```

Out[8]:

```
array([[ 4806,  6336,  7866],
       [ 5994,  7902,  9810],
       [ 7182,  9468, 11754]])
```

3) Distributive [A(B+C) = AB + AC]

In [9]:

```
A.dot(B + C)
```

Out[9]:

```
array([[ 588,  660,  732],
       [ 732,  822,  912],
       [ 876,  984, 1092]])
```

In [10]:

```
(A.dot(B)) + (A.dot(C))
```

Out[10]:

```
array([[ 588,  660,  732],
       [ 732,  822,  912],
       [ 876,  984, 1092]])
```

4) Multiplicative identity (AI = IA)

In [11]:

```
D = np.random.randint(0, 5, (5, 5))
```

```
D
```

Out[11]:

```
array([[4, 1, 2, 3, 2],
       [4, 0, 3, 2, 0],
       [2, 1, 1, 3, 3],
       [0, 3, 3, 3, 4],
       [3, 0, 2, 4, 2]])
```

In [12]:

```
I = np.identity(5)
I
```

Out[12]:

```
array([[1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1.]])
```

In [13]:

```
np.dot(D, I)
```

Out[13]:

```
array([[4., 1., 2., 3., 2.],
       [4., 0., 3., 2., 0.],
       [2., 1., 1., 3., 3.],
       [0., 3., 3., 3., 4.],
       [3., 0., 2., 4., 2.]])
```

In [14]:

```
np.dot(I, D)
```

Out[14]:

```
array([[4., 1., 2., 3., 2.],
       [4., 0., 3., 2., 0.],
       [2., 1., 1., 3., 3.],
       [0., 3., 3., 3., 4.],
       [3., 0., 2., 4., 2.]])
```

Task 2 :- Matrix Inverse

In [15]:

```
mat = np.random.randint(0,100,(5,5))
mat
```

Out[15]:

```
array([[ 2, 81, 77, 95, 73],
       [20,  1, 28, 31, 85],
       [ 7, 60, 44, 50, 85],
       [41, 28, 98, 92, 40],
       [58, 41, 81, 29,  3]])
```

In [16]:

```
mat_invert = np.linalg.inv(mat)
```

```
mat_invert
```

Out[16]:

```
array([[-0.46067861, -0.32902508,  0.58029919,  0.31789376, -0.14817034],
       [-0.13567775, -0.11267002,  0.18900132,  0.0885485 , -0.04187486],
       [ 0.51300488,  0.37901528, -0.65757374, -0.35810631,  0.18412171],
       [-0.32589015, -0.24705471,  0.41652588,  0.24410307, -0.12639712],
       [ 0.05985539,  0.05575811, -0.07406144, -0.04690168,  0.02080217]])
```

Task 3 :- Numpy v/s Loops (Time computation)

In [17]:

```
matrix1 = np.random.randint(5, size=(10000, 10000))
matrix2 = np.random.randint(5, size=(10000, 10000))
```

In [18]:

```
#Using For Loops

start = time.time()
for i in range(len(matrix1)):
    for j in range(len(matrix1)):
        matrix1[i][j] = matrix1[i][j] + 1

print("Time taken using loops:")
print(time.time() - start)
```

```
Time taken using loops:
198.81241416931152
```

In [19]:

```
#Using Numpy

start = time.time()
ans = np.add(matrix2, matrix2)
print("Time taken using numpy:")
print(time.time() - start)
```

```
Time taken using numpy:
0.2942678928375244
```