Matrix Operation using Numpy

February 20, 2025

1 Matrix Operations Using Numpy

[ ]:

```
import numpy as np
```

Create Matrix in NumPy

In NumPy, we use the np.array() function to create a matrix. For example,

[ ]:

```
#create a 2x2 matrix
matrix1 = np.array([[1, 3],
                    [5, 7]])
print("Output\n")
print("2x2 Matrix:\n",matrix1)

# create a 3x3  matrix
matrix2 = np.array([[2, 3, 5],
                          [7, 14, 21],
                     [1, 3, 5]])

print("\n3x3 Matrix:\n",matrix2)
```

```
Output

2x2 Matrix:
 [[1 3]
 [5 7]]

3x3 Matrix:
 [[ 2  3  5]
 [ 7 14 21]
 [ 1  3  5]]
```

Perform Matrix Multiplication in NumPy

We use the np.dot() function to perform multiplication between two matrices. For example,

[ ]:

```
# create two matrices
matrix1 = np.array([[1, 3],
                              [5, 7]])
```

```

matrix2 = np.array([[2, 6],
                    [4, 8]])

# calculate the dot product of the two matrices
result = np.dot(matrix1, matrix2)
print("Output\n")
print("matrix1 x matrix2: \n",result)
```

```
Output

matrix1 x matrix2:
 [[14 30]
 [38 86]]
```

Or we can also do this by np.matmul() or simply as x@y

[ ]:

```
#Matrix Multiplication
x=np.array([[1,2,3],[4,5,6]])
y=np.array([[7,8,9],[10,11,12],[13,14,15]])
z=np.matmul(x,y)
w=x@y
print(z,w)
```

```
[[ 66  72  78]
 [156 171 186]] [[ 66  72  78]
 [156 171 186]]
```

Transpose NumPy Matrix

The transpose of matrix is a new matrix that is obtained by exchanging the rows and columns. For 2x2 matrix,

Matrix:

a11 a12

a21 a22

Transposed Matrix:

a11 a21

a12 a22

In NumPy, we can obtain the transpose of a matrix using the np.transpose() function. For example,

[ ]:

```
# create a matrix
matrix1 = np.array([[1, 3],
                             [5, 7]])

# get transpose of matrix1
```

```
result = np.transpose(matrix1)

print("Output\n")
print(result)
```

```
Output

[[1 5]
 [3 7]]
```

Calculate Inverse of a Matrix in NumPy

In NumPy, we use the np.linalg.inv() function to calculate the inverse of the given matrix.

However, it is important to note that not all matrices have an inverse. Only square matrices that have a non-zero determinant have an inverse.

Now, let’s use np.linalg.inv() to calculate the inverse of a square matrix.

[ ]:

```
# create a 3x3 square matrix
matrix1 = np.array([[1, 3, 5],
                             [7, 9, 2],
                    [4, 6, 8]])

# find inverse of matrix1
result = np.linalg.inv(matrix1)

print("Output\n")
print(result)
```

```
Output

[[-1.11111111 -0.11111111  0.72222222]
 [ 0.88888889  0.22222222 -0.61111111]
 [-0.11111111 -0.11111111  0.22222222]]
```

Find Determinant of a Matrix in NumPy

We can find the determinant of a square matrix using the np.linalg.det() function to calculate the determinant of the given matrix.

Suppose we have a 2x2 matrix A:

a b

c d

So, the determinant of a 2x2 matrix will be:

det(A) = ad - bc where a, b, c, and d are the elements of the matrix.

[ ]:

```
# create a matrix
matrix1 = np.array([[1, 2, 3],
                             [4, 5, 1],
                    [2, 3, 4]])

# find determinant of matrix1
result = np.linalg.det(matrix1)

print("Output\n")
print(result)
```

```
Output

-5.000000000000001
```

Flatten Matrix in NumPy

Flattening a matrix simply means converting a matrix into a 1D array.

To flatten a matrix into a 1-D array we use the array.flatten() function. Let's see an example.

[ ]:

```
# create a 2x3 matrix
matrix1 = np.array([[1, 2, 3],
                             [4, 5, 7]])

result = matrix1.flatten()

print("Output\n")
print("Flattened 2x3 matrix:", result)
```

```
Output

Flattened 2x3 matrix: [1 2 3 4 5 7]
```

Fetching Distinct Values from an array

[ ]:

```
#Fetching Distinct Values from an array
x=np.array([7,8,9,3,3,4,5,2,2,6,7,2,6,8,1,2,0])
print(np.unique(x))
```

```
[0 1 2 3 4 5 6 7 8 9]
```

Finding largest element from an array

[ ]:

```
#Finding largest element from an array
x=np.array([85,96,25,36,14,852,6548,1,2,5,6,9])
index=np.argmax(x)
print(index)
print("The Largest element in the array \n",x,"\nis",x[index],"at index",index)
```

```
6
The Largest element in the array
```

```
 [  85   96   25   36   14  852 6548    1    2    5    6    9]
is 6548 at index 6
```

Finding smallest element from an array

[ ]:

```
#Finding smallest element from an array
x=np.array([85,96,25,36,14,852,6548,1,2,5,6,9])
index=np.argmin(x)
print(index)
print("The Smallest element in the array \n",x,"\nis",x[index],"at index",index)
```

```
7
The Smallest element in the array
 [  85   96   25   36   14  852 6548    1    2    5    6    9]
is 1 at index 7
```

Finding indices that would sort the array

[ ]:

```
#Finding indices that would sort the array
x=np.array([85,96,25,36,14,852,6548,1,2,5,6,9])
index=x.argsort()
print(index)
```

```
[ 7  8  9 10 11  4  2  3  0  1  5  6]
```