Experiment No. 1
Implementation of Matrix operations and data representation towards understanding mathematics for recommendation system
Date of Performance:
Date of Submission:
Marks:
Sign:

Aim: Implementation of Matrix operations and data representation towards understanding mathematics for recommendation system

Objective: Ability to understand the different matrix operations performed using Python respectively.

Theory: The matrix operations help us to combine two of more matrices, to form a single matrix. The arithmetic operations of addition, subtraction, multiplication can also be performed on matrices. Further sometimes matrix operations can change a matrix itself. The operation of the transpose of matrix, the inverse of matrix helps to transform a specific matrix onto itself. The following are some of the important matrix operations. Addition - matrix operations, Subtraction - matrix operations, Multiplication - matrix operations, Transpose operation of a matrix and Inverse operation of a matrix.

Addition of Matrices

As we add two numbers we can easily add two matrices. The only thing we have to note is that the order of both the matrices that are to be added must be the same. That is to add two matrices we have to make sure that they are of the same order and then each element of the first matrix adds with each element of the second matrix to get a single matrix and thus the addition operation gets completed.

Subtraction of Matrices

As we add two matrices e can also easily subtract two matrices. The only thing we have to note is that the order of both the matrices that are to be subtracted must be the same. That is to subtract two matrices we have to make sure that they are of the same order and then each element of the first matrix is subtracted with each element of the second matrix to get a single matrix and thus the subtraction operation gets completed.

Scalar Multiplication of Matrices

For any matrix A = [aij]m×n if we multiply the matrix A with any scaler (say k) then the scaler is multiplied by each element of the matrix and this is called the scalar multiplication of matrices.

Multiplication of Matrix

Matrix multiplication is the operation that helps us to multiply two matrices. This is different from algebraic multiplication and not all the matrices can be multiplied. Only those matrices can be multiplied where the number of columns in the first is equal to the number of rows in the second, i.e for matrix Am×n and matrix Bn×p the multiplication is possible for any other matrices where the column of the first matrix is not equal to the row in the second matrix the multiplication is not possible.

Transpose Operation of a Matrix

Tramspose operation of a matrix is used to find the transpose of any matrix. Transpose of any matrix is a matrix in which the rows of the matrix are changed to the column of the matrix and the column of the matrix is changed to the rows of the matrix. Suppose we have a matrix A of order m×n such that A =[ij]m×n then the transpose of matrix A is represented as (A)T

Inverse Operation of a Matrix

For any matrix A its inverse is found only when A is a square matrix and its determinant is equal to 1.

Implementation:

```
import numpy as np
```

Declaring matrices

```
mx1 = np.array([[5, 10], [15, 20]])
mx2 = np.array([[25, 30], [35, 40]])

print("Matrix1 =\n",mx1)
print("\nMatrix2 =\n",mx2)
```

```
Matrix1 =
 [[ 5 10]
 [15 20]]
```

```
Matrix2 =
 [[25 30]
 [35 40]]
```

Addition using Numpy methods

```
print ("\nAddition of two matrices: ")
print (np.add(mx1,mx2))
```

```
Addition of two matrices:
[[30 40]
 [50 60]]
```

Subtraction using Numpy methods

```
print ("\nSubtraction of two matrices: ")
print (np.subtract(mx1,mx2))
```

```
Subtraction of two matrices:
[[-20 -20]
 [-20 -20]]
```

Division using Numpy methods

```
print ("\nMatrix Division: ")
print (np.divide(mx1,mx2))
```

```
Matrix Division:
[[0.2        0.33333333]
 [0.42857143 0.5       ]]
```

Multiplication using Numpy methods

```
print ("\nMultiplication of two matrices: ")
print (np.multiply(mx1,mx2))
```

```
Multiplication of two matrices:
[[125 300]
 [525 800]]
```

```
mx1 @ mx2
```

```
array([[ 475,  550],
       [1075, 1250]])
```

```
np.matmul (mx1, mx2)
```

```
array([[ 475,  550],
       [1075, 1250]])
```

```
np.dot (mx1, mx2)
```

```
array([[ 475,  550],
       [1075, 1250]])
```

```
mx = np.array([[5, 10], [15, 20]])

print("Matrix =\n",mx)

print ("\nThe summation of elements=")
print (np.sum(mx))

print ("\nThe column wise summation=")
print (np.sum(mx,axis=0))

print ("\nThe row wise summation=")
print (np.sum(mx,axis=1))
```

```
Matrix =
 [[ 5 10]
 [15 20]]

The summation of elements=
50

The column wise summation=
[20 30]

The row wise summation=
[15 35]
```

```
mx = np.array([[5, 10], [15, 20]])

print("Matrix =\n",mx)

print ("\nThe Transpose =")
print (mx.T)
```

```
Matrix =
 [[ 5 10]
 [15 20]]

The Transpose =
[[ 5 15]
 [10 20]]
```

Numpy method is also available

```
np.transpose (mx)
```

```
array([[ 5, 15],
       [10, 20]])
```

Inverse of a matrix

```
np.linalg.inv (mx)
```

```
array([[-0.4,  0.2],
       [ 0.3, -0.1]])
```

Conclusion:

The implementation of Matrix operations and data representation fosters deeper mathematical comprehension. Utilizing these tools enhances understanding of mathematical concepts, aiding recommendation systems in providing more accurate and insightful suggestions. In conclusion, embracing these techniques empowers users to engage more meaningfully with mathematics, enriching their learning experience.