Data Manipulation

- In order to get anything done, we need some way to store and manipulate data.

There are two important things we need to do with data:

(i)Acquire them

(ii)Process them once they are inside the computer.

Python for Data manipulation

- Let us consider synthetic data,we introduce the n-dimensional array, which is also called the *tensor.*
- We import the np (numpy)npx (numpy_extension) modules from MXNet(MXNet is an open-source deep learning software framework, used to train, and deploy deep neural networks.)
- Here,the np module includes functions supported by NumPy, while the **npx module contains a set of extensions developed to empower deep learning within a NumPy-like environment.**
- When using tensors,we almost always invoke the set_np function: this is for compatibility of tensor processing by other components of MXNet.

Python for Data manipulation…

```
# Import the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
x=[1,2,3,4,5,6,7,8]
X
o/p[1, 2, 3, 4, 5, 6, 7, 8]
```

Python for Data manipulation..

```
x = np.arange(14)
```

X

o/p

```
array([ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
13])
```

```
x.shape
```

```
(14)
```

```
y=x.size
```

```
y
```

14

Python for Data manipulation..

Change the shape of a tensor without altering either the number of elements or their values

```
a =x.reshape(3,4) #Arrange the array of size 12 into 3 rows and 4 columns

a

o/p

array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

Python for Data manipulation..

- Create a set of two arrays with Zeros

```
np.zeros((2, 3, 4))
#print two arrays with 3rows and four columns with values zero.
O/P
array([[[0., 0., 0., 0.],
        [0., 0., 0., 0.],
        [0., 0., 0., 0.]],

       [[0., 0., 0., 0.],
        [0., 0., 0., 0.],
        [0., 0., 0., 0.]]])
```

Python for Data manipulation..

- np.ones((2, 3, 4))
- `#print two arrays with 3rows and four columns with values ones.`

o/p

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

Python for Data manipulation..

- Construct an array-Random values
- For example,when we construct arrays to serve as parameters in a neural network.
- We willt ypically initialize their values randomly. The following snippet creates a tensor with shape (3,4).
- Each of its elements is randomly sampled from a standard Gaussian(normal)distribution with a mean of 0 and a standard deviation of 1.

Python for Data manipulation..

np.random.normal(0, 1, size=(3, 4))

o/p

```
array([[ 0.30643967, 1.3812342 , -0.57935724, -
0.65810456], [-1.47102494, -0.62923998, -
1.5654041 , 1.47659725], [ 0.45809148, -
0.4619724 , -0.2459706 , 0.81116768]])
```

Python for Data manipulation..

- We can also specify the exact values for each element in the desired tensor by supplying a Python list(or list of lists)containing the numerical values.
- np.array([[2, 1, 4, 3], [1, 2, 3, 4], [4, 3, 2, 1]])

o/p

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

Arithmetic Operations

```
x = np.array([1, 2, 4, 8])
y = np.array([2, 2, 2, 2])
x + y, x - y, x * y, x / y, x ** y
# The ** operator is exponentiation
o/p
(array([ 3, 4, 6, 10]),
array([-1, 0, 2, 6]),
array([ 2, 4, 8, 16]),
array([0.5, 1. , 2. , 4. ]),
array([ 1, 4, 16, 64]))
```

Arithmetic Operations …

- `np.exp(x)`
- `o/p`
- `array([2.71828183e+00, 7.38905610e+00, 5.45981500e+01, 2.98095799e+03])`

Arithmetic Operations …

```
X = np.arange(12).reshape(3, 4)
Y = np.array([[2, 1, 4, 3], [1, 2, 3, 4], [4, 3,
 2, 1]])
np.concatenate([X, Y], axis=0),
np.concatenate([X, Y], axis=1)
o/p
array([[ 0, 1, 2, 3, 2, 1, 4, 3], [ 4, 5, 6, 7,
1, 2, 3, 4], [ 8, 9, 10, 11, 4, 3, 2, 1]])
```

Arithmetic Operations …

- `X == Y`

```
array([[False,  True, False,  True],
       [False, False, False, False],
       [False, False, False, False]])
```

- `X.sum()`
- 66

Arithmetic Operations …

- We saw how to perform element wise operations on two tensors of the same shape.
- when shapes differ,we can still perform element wise operations by invoking the broadcasting mechanism.
- First, expand one or both arrays by copying elements appropriately so that after this transformation , the two tensors have the same shape.

Arithmetic Operations …

```
# Import the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

- a = np.arange(3).reshape(3, 1)
- b = np.arange(2).reshape(1, 2)
- a, b
- o/p
- (array([[0.],
- [1.],
- [2.]]),
- array([[0., 1.]]))

Arithmetic Operations …

- a and b are 3×1 and 1×2 matrices respectively, their shapes do not match up if we want to add them.
- We broadcast the entries of both matrices into a larger 3×2 matrix as follows
- for matrix a it replicates the columns and for matrix b it replicates the rows before adding up both element wise.
- a + b
- o/p
- array([[0., 1.],
- [1., 2.],
- [2., 3.]])

Indexing and Slicing

- X=[1,2,3,4,5,6]
- X[-1], X[1:3]

#[-1] selects the Last element and[1:3]selects the second and the third elements as follows:

- o/p

6

3,4

- X[1, 2] = 3

Saving Memory

Y = id(X)

- Y
- O/P
- 140064807278144 #Memory Location of X

Conversion to Other Python Objects

- a = np.array([3.5]) a,
- a.item(), float(a), int(a)
- (array([3.5]), 3.5, 3.5, 3)

Practice questions

- W.A.P to Change the shape of a tensor without altering either the number of elements or their values.
- W.A.P to print two arrays with 3rows and four columns with values zero.
- W.A.P to print two arrays with 3rows and four columns with values ones.
- W.A.P to print a random array of size(3,4)
- What is Indexing and slicing
- Write the function for Converting and object from one form to Other Python Objects.Write an application in which conversion is required.