SONET

Data Science
(Level-I)

Ajeet K. Jain, M. Narsimlu
(ML TEAM)- SONET, KMIT, Hyderabad

Session - 18

This session deals with

Numpy

Numpy Operations

Matrix and its operations

Linear algebra

System of equations

Accessing Elements

Multi-Dimensional Arrays

```
In [38]: a=arange(27).reshape(3,3,3)
         print(a)
         print(a[1:3,1:3,1:3])

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

 [[ 9 10 11]
  [12 13 14]
  [15 16 17]]

 [[18 19 20]
  [21 22 23]
  [24 25 26]]]
[[[13 14]
  [16 17]]

 [[22 23]
  [25 26]]]
```

Attributes of numpy

Various attributes are defined in numpy.

```
In [68]: a=array([[1,2,3],[4,5,6]])
         print(a.shape)
         print(a.strides)
         print(a.ndim)
         print(a.data)
         print(a.size)
         print(a.itemsize)
         print(a.nbytes)
         print(a.flags)

(2, 3)
(12, 4)
2
<memory at 0x00000199D73F28B8>
6
4
24
  C_CONTIGUOUS : True
  F_CONTIGUOUS : False
  OWNDATA : True
  WRITEABLE : True
  ALIGNED : True
  UPDATEIFCOPY : False
```

```
In [70]: a=array([[1,2,3],[4,5,6]])
         print(a.T)
         print(a.dtype)

[[1 4]
 [2 5]
 [3 6]]
int32
```

Arithmetic Operations

In [109]:

```
a=array([[1,2],[4,5]])
b=array([[1,0],[0,1]])
print(a)
print(b)
print("........a+b.........")
c=a+b
print(c)
print("........a-b.........")
c=a-b
print(c)
print("........a*b.........")
c=a*b
print(c)
```

```
[[1 2]
 [4 5]]
[[1 0]
 [0 1]]
........a+b.........
[[2 2]
 [4 6]]
........a-b.........
[[0 2]
 [4 4]]
........a*b.........
[[1 0]
 [0 5]]
```

Methods on numpy

```
In [92]: a=array([[0,2,3],[4,5,6]])
         print(a)
         print(a.sum())
         print(a.sum(1))
         print(a.sum(0))
         print(a.max())
         print(a.max(1))
         print(a.min())
         print(a.mean())
         print(a.var())
         print(a.all())
         print(a.any())
```

```
[[0 2 3]
 [4 5 6]]
20
[ 5 15]
[4 7 9]
6
[3 6]
0
3.33333333333
3.88888888889
False
True
```

Methods on numpy

Various methods defined on numpy.

```
In [56]:  a=arange(9)
          b=a.reshape(3,3)
          print(b)

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

```
In [97]:  a=array([[1,2],[4,5]])
          b=array([[1,0],[0,1]])
          print(a)
          print(b)
          c=a.dot(b)
          print(c)

[[1 2]
 [4 5]]
[[1 0]
 [0 1]]
[[1 2]
 [4 5]]
```

```
In [100]:  a=array([30,12,23,54,5,36])
           print(a)
           print(sort(a))

[30 12 23 54  5 36]
[ 5 12 23 30 36 54]
```

```
In [107]:  a=array([[0,2,3],[4,5,6]])
           print(a)
           print(a.tolist())
           print(a.transpose())

[[0 2 3]
 [4 5 6]]
[[0, 2, 3], [4, 5, 6]]
[[0 4]
 [2 5]
 [3 6]]
```

Strides

Number of bytes to be skipped to get next element

In [29]:
```
a=arange(9)
print(a)
print(a.strides)
```

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

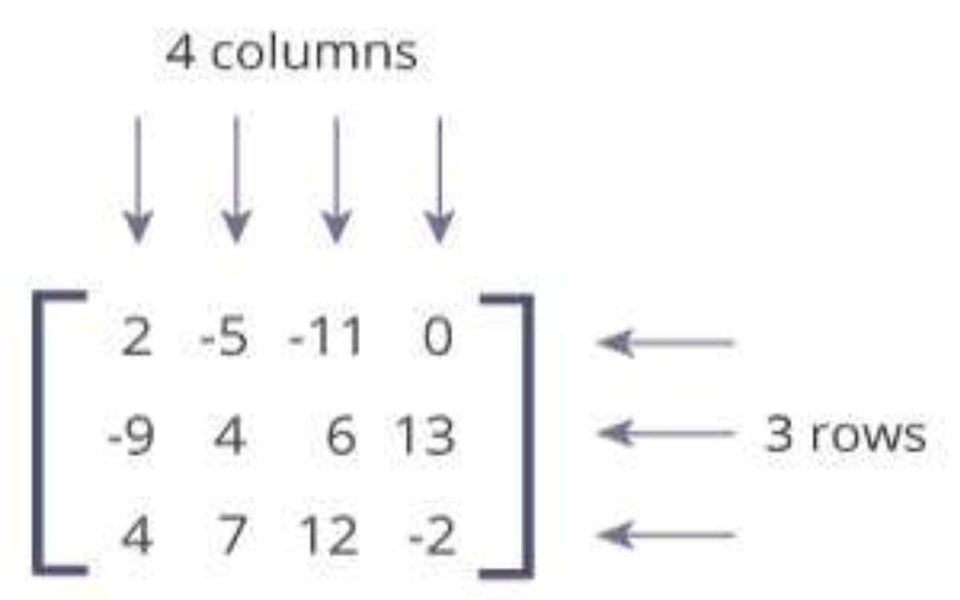
In [37]:
```
a=arange(9).reshape(3,3)
print(a)
print(a.dtype)
print(a.strides)
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
int32
(12, 4)
```

Total 12 bytes
4 bytes each element

Matrix

A matrix is a two-dimensional data structure where numbers are arranged into rows and columns.



This matrix is a 3x4 (pronounced "three by four") matrix because it has 3 rows and 4 columns.

Matrix

A python matrix is a list of list with 3 rows and 4 columns.

```
import numpy as np
A = [[1, 4, 5, 12],
     [-5, 8, 9, 0],
     [-6, 7, 11, 19]]
m1=np.matrix(A)
print(m1)
```

Output

```
[[ 1  4  5 12]
 [-5  8  9  0]
 [-6  7 11 19]]
```

Accessing Elements

A python matrix is a list of list with 3 rows and 4 columns.

```
import numpy as np
A = [[1, 4, 5, 12],
     [-5, 8, 9, 0],
     [-6, 7, 11, 19]]
m1=np.matrix(A)
#  first row
print("m1[0]=", m1[0])
# second row
print("m1[1] =", m1[1])
# last row
print("m1[-1] =", m1[2])
```

Output

```
m1[0]= [[ 1  4  5 12]]
m1[1] = [[-5  8  9  0]]
m1[-1] = [[-6  7 11 19]]
```

Accessing Elements

DATA SCIENCE

A python matrix is a list of list with 3 rows and 4 columns.

```
import numpy as np
A = [[1, 4, 5, 12],
    [-5, 8, 9, 0],
    [-6, 7, 11, 19]]
m1=np.matrix(A)
print("m1[:,0] =",m1[:,0]) # First Column
print("m1[:,3] =", m1[:,3]) # Fourth Column
print("m1[:,-1] =", m1[:,-1]) # Last Column (4th column in this case)
```

Output

```
m1[:,0] = [[ 1]
 [-5]
 [-6]]
m1[:,3] = [[12]
 [ 0]
 [19]]
m1[:,-1] = [[12]
 [ 0]
 [19]]
```

Matrix - Operations

It can perform addition, multiplication etc….

```
import numpy as np
A = np.matrix([[2, 4], [5, -6]])
B = np.matrix([[9, -3], [3, 6]])
C = A + B        # element wise addition
print(C)
```

Output

```
[[11  1]
 [ 8  0]]
```

Matrix - Operations

Multiplication

```
import numpy as np
A = np.matrix([[2, 4], [5, -6]])
B = np.matrix([[9, -3], [3, 6]])
C = np.dot(A,B)
print(C)
```

Output

```
[[ 30  18]
 [ 27 -51]]
```

Matrix - Operations

Transpose

```
import numpy as np
A = np.matrix([[2, 4], [5, -6]])
B = np.matrix([[9, -3], [3, 6]])
print(A.transpose())
```

Output

```
[[ 2  5]
 [ 4 -6]]
```

Matrix - Operations

Using numpy we can perform several matrix operations such as determinant, inverse, rank of matrix etc…

```
import numpy as np
A = np.matrix([[2, 4], [5, -6]])
print(np.linalg.det(A))
print(np.linalg.inv(A))
print(np.linalg.matrix_rank(A))
```

Output

```
-32.0
[[ 0.1875   0.125  ]
 [ 0.15625 -0.0625 ]]
2
```

Linear Algebra

More common problems in linear algebra is solving a matrix-vector equation.

Linear Algebra as one of the foundational blocks of Data Science.

Linear algebra is a sub-field of mathematics concerned with vectors, matrices, and linear transforms.

Linear Algebra

$$A\mathbf{x} = \mathbf{b}$$

$$A = \begin{bmatrix} 2 & 1 & -2 \\ 3 & 0 & 1 \\ 1 & 1 & -1 \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} -3 \\ 5 \\ -2 \end{bmatrix}$$

Will constructing the arrays for *A* and **b for solving x**.

```
A = np.array([[2,1,-2],[3,0,1],[1,1,-1]])
b = np.transpose(np.array([[-3,5,-2]])
```

To solve the system we do

```
x = np.linalg.solve(A,b)
```

Why Linear Algebra

The use of linear algebra structures when working with data, such as tabular datasets and images.

Linear algebra concepts when working with data preparation, such as one hot encoding and dimensionality reduction.

The ingrained use of linear algebra notation and methods in sub-fields such as deep learning, natural language processing, and recommender systems.

SONET

Usage of Linear Algebra

DATA SCIENCE

In a multiple regression problem we seek a function that can map input data points to outcome values..

Each data point is a *feature vector* (x_1 , x_2 , …, x_m) composed of two or more data values that capture various features of the input.

To represent all of the input data along with the vector of output values we set up a input matrix X and an output vector y:

Usage of Linear Algebra

$$X = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,m} \\ 1 & x_{2,1} & x_{2,2} & \cdots & x_{2,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & x_{n,2} & \cdots & x_{n,m} \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

we can compute a predicted outcome value

$$\hat{y} = \mathbf{x} \cdot \boldsymbol{\beta} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_m x_m$$

Usage of Linear Algebra

To compute the **β** vector as follows:

$$\boldsymbol{\beta} = (X^T X)^{-1} X^T \mathbf{y}$$

will use numpy to construct the appropriate matrices and vectors and solve for the β vector.

Once we have solved for β we will use it to make predictions for some test data points that we initially left out of our input data set.

Usage of Linear Algebra

constructed the input matrix X and the outcomes vector y in numpy, the following code will compute the β vector:

```
Xt = np.transpose(X)
XtX = np.dot(Xt,X)
Xty = np.dot(Xt,y)
beta = np.linalg.solve(XtX,Xty)
```

Solving System equations

SONET

Solving System equations

DATA SCIENCE

$$3x + 7y = 27$$
$$5x + 2y = 16$$

This can be put in the matrix dot product form as

$$\begin{bmatrix} 3 & 7 \\ 5 & 2 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 27 \\ 16 \end{bmatrix}$$

If A represents the matrix of coefficients, x the column vector of variables and B the column vector of solutions, the above equation can be shortened to

$$Ax = B$$

Solution

```
import numpy as np
a = np.array([[3,7], [5,2]])
b = np.array([27,16])
x = np.linalg.solve(a, b)
print(x)
```

Output

```
[2. 3.]
```

Conclusion

You are aware of

Dictionary Data Structures

Numpy

We will proceed with

More on Numpy

THANK YOU