Numpy Fundamentals

- **Date:** November 16, 2023
- **Author:** Engr Muhammad Saqlain
- **Institute:** Department of Computer System Engineering, IUB

```
# import numpy
import numpy as np
```

Flattening Arrays

```
x = np.arange(15).reshape(5,3)
x

x.shape

y = x.ravel()
y

y.shape

x

y[0]=99

y

x
```

Ravel method does not produce acopy We use flatten method which produces copy

```
x

z = x.flatten()
z

z[0] = -100

z

x
```

Concatenating Arrays

```
arr1 = np.arange(1,7).reshape(2,3)
arr1

arr2 = np.arange(7,13).reshape(2,3)
arr2
```

```
np.concatenate([arr1,arr2], axis=0)

np.concatenate([arr1,arr2], axis=1)
```

Stacking

```
arr1

arr2

np.vstack((arr1,arr2))

np.row_stack((arr1,arr2))

np.hstack((arr1,arr2))

np.column_stack((arr1,arr2))
```

Stacking lower order arrays to create higher order arrays

```
import numpy as np
a1 = np.arange(1,13)
a1

a1.shape

a2 = np.arange(13,25)
a2

a2.shape

a3 = np.stack((a1,a2), axis=0)  # column-wise stacling of elements
a3

a3.shape

a4 = np.stack((a1,a2), axis=1) # row-wise stacking of elements
a4

a4.shape

x1 = np.arange(1,13).reshape(3,4)
x1

x1.shape

x2 = np.arange(13,25).reshape(3,4)
x2

x2.shape

x3 = np.stack((x1,x2), axis=0)
x3
```

```
x3.shape

x4 = np.stack((x1,x2), axis=1)
x4

x4.shape

x5 = np.stack((x1,x2), axis=2)
x5

x5.shape
```

Linear Algebra

Dot product

```
a = np.arange(1,5)
a

b = np.arange(5,9)
b

np.dot(a,b)

a@b

x = np.arange(1,7).reshape(2,3)
x

y = np.ones(3)
y

x@y

p = np.random.randint(0,10,(4,4))
p

q = np.random.randint(-5,5,(4,4))
q

p@q   # normal matrix multiplication
```

Inverse of a matrix

```
np.linalg.inv(p)
```

QR Decomposition of a matrix

```
q,r = np.linalg.qr(p)  # used in curve fitting or regression (least
squares)
```

```
q  # orthogonal matrix i.e. its transpose is equal to its inverse

q.T

np.linalg.inv(q)

r

q@r

p
```

Diagonal and trace

```
p

np.diag(p)

np.trace(p)
```

Eigenvalues and eigenvectors of a square matrix

```
p

w, v = np.linalg.eig(p)

w

w.shape

v

v.shape
```

Singular value decomposition

```
p

L, S, R = np.linalg.svd(p)

L

S

R

L@(np.diag(S))@R
```

Solving Linear Systems of Equations

\begin{equation} 3x-y-z=0\ x+y=5\ 2x-3z=2 \end{equation}

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

b=np.array([[0],[5],[2]])
b

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

Computing norms

```
x = np.array([[0,3,4],[2,6,4]])
x

np.linalg.norm(x)  # norm of all glgmgnts

np.linalg.norm(x,axis=0)  # column-wis norm

np.linalg.norm(x,axis=1)  # row-wis norm
```