NumPy Tutorial

Abhinav Gopal, Shrey Vasavada

INTRODUCTION | NumPy

- NumPy is the go-to library for linear algebra operations using python
- Heavily used in machine learning, quantum computing, and data analytics
- Has quick, optimized linear algebra datatypes and methods
- Easy to Use
- Adapts to different hardware

Linear Algebra Problem Solving | NumPy

- Matrix operations: addition, subtraction, multiplication, inversion, power
- Matrix Decompositions: Diagonalization (using eigenvalues and eigenvectors), Singular Value Decomposition (To be learned later in this course)
- Equation Solvers: Linear Regression solver (Least Squares)

Basics | NumPy

- Vector creation, matrix creation (Do it using python lists!)
- ```
  lst = [3,6,9,10]
  arr = np.array([3,6,9,10])
  type(lst), type(arr)
  ```
- ```
  arr_2D = np.array([[3,6],
           [9,10]])
  arr_2D, arr_2D.shape
  ```
  ```
  (array([[ 3,  6],
          [ 9, 10]]), (2, 2))
  ```

Basics | NumPy

- Appending NumPy arrays
-
```
arr = np.array([3,6,9,10])
arr = np.append(arr, 1)
arr
```
```
array([ 3,  6,  9, 10,  1])
```

-
```
arr_2D = arr.reshape(2,4)
arr_2D = np.append(arr_2D, [[3,4,10,500]], axis = 0)
arr_2D
```
```
array([[  3,   6,   9,  10],
       [  1,   8,   0,  11],
       [  3,   4,  10, 500]])
```

Stacking NumPy arrays

```
arr1 = np.array([1,2,3])
arr2 = np.array([4,5,6])
arr_1D = np.hstack((arr1, arr2))
arr_1D
```

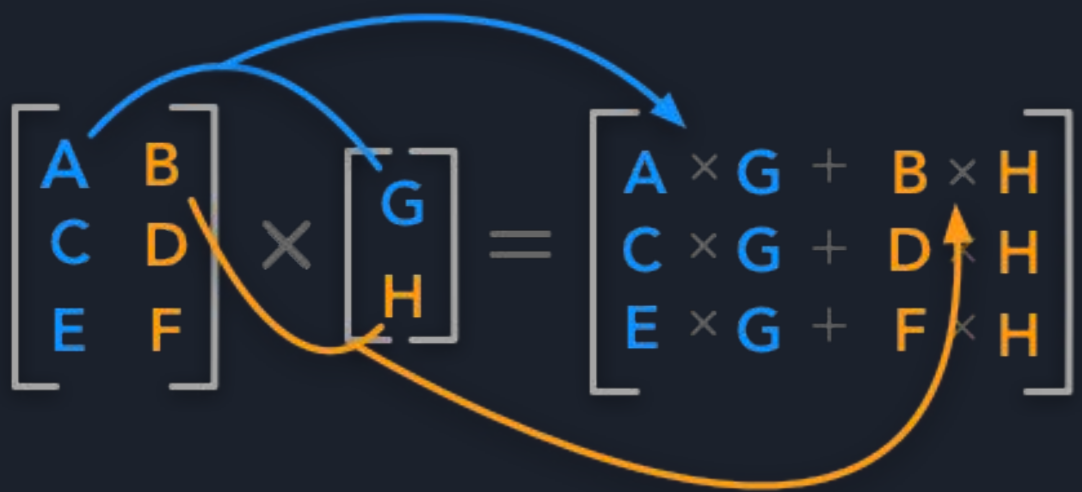
```
array([1, 2, 3, 4, 5, 6])
```

```
arr_2D = np.vstack((arr1, arr2))
arr_2D
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

Benefits| NumPy

- NumPy's primary benefit comes from the speed with which linear algebra calculations can be performed
- Faster operations allow us to do calculations on larger matrices and arrays
- Recall matrix multiplication:



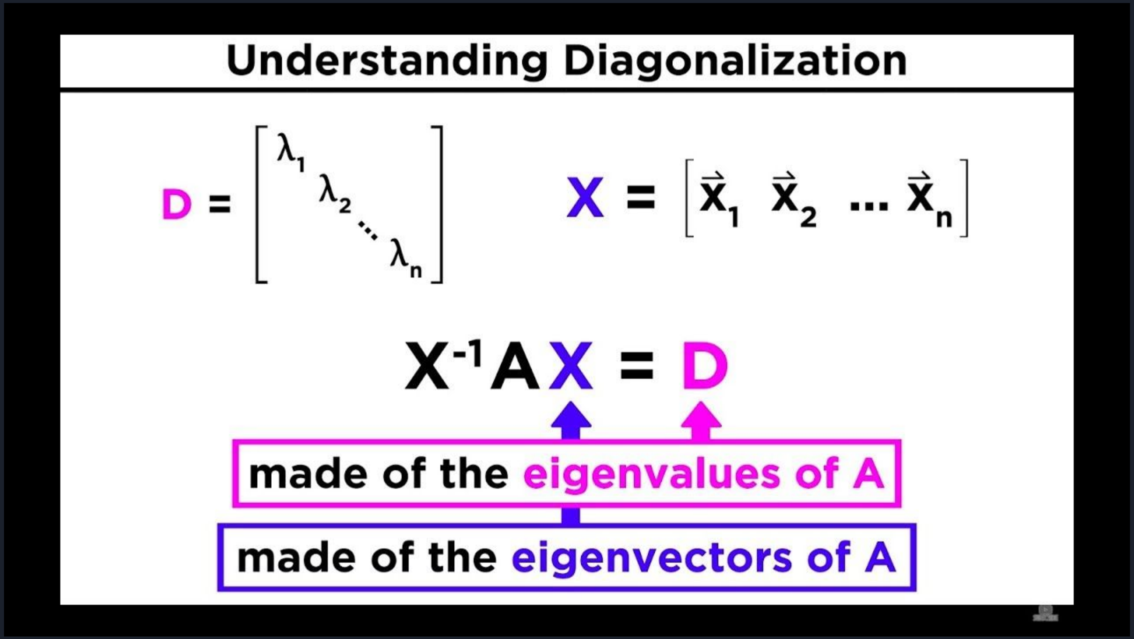
np.einsum| NumPy

- Combines several NumPy operations into 1.
- First argument: String specifying the math operation
- Second argument: The matrix itself.
-

```
# transpose of mat1
mat1 = np.array([1,2,3,4,5,6]).reshape(2,3)
np.einsum('ji', mat1)
```

```
array([[1, 4],
       [2, 5],
       [3, 6]])
```

Diagonalization of matrices



https://www.youtube.com/watch?v=WTLl03D4TNA

Eigenvectors and Eigenvalues | NumPy

- To compute eigenvectors and eigenvalues, we can simply use np.linalg.eig(matrix).
- The operation returns a vector of eigenvalues, and a matrix of eigenvectors, where the column vectors are the eigenvectors.

```
A = np.array([[1,2],[3,4]])
print(np.linalg.eig(A))
```

```
(array([-0.37228132,  5.37228132]), array([[-0.82456484, -0.41597356],
       [ 0.56576746, -0.90937671]]))
```

np.lstsq | NumPy

- Does the classic least squares operation using NumPy.
- Accepts the X matrix and the y vector as the parameters
- The first returned value from that function is the estimate of the coefficients.
-

```
X = np.array([[2,2,3,1], [4,1,1,2],[1,9,0, 1],[6,1,0,10], [6,1,5,3]])
y = np.array([10.75, 11, 4.25, 214, 20.75])
w = np.linalg.lstsq(X,y)[0]
print(w)
```

```
[-12.31108269  -1.31390695   2.19785191  28.91426871]
```