5CS037 Concepts and Technologies of AI
Workshop -1
Numpy For Matrix Manipulation.

Siman Giri

November 14, 2023

1. Numpy: Introduction.

1.1 What is Numpy?

- Numpy is an open-source add-on modules to python which provides common mathematical and numerical routines in pre-compiled, fast functions.

1.1 What is Numpy?

- Numpy is an open-source add-on modules to python which provides common mathematical and numerical routines in pre-compiled, fast functions.
- The NumPy(Numeric Python) package provides basic routines for manipulating large arrays and matrices of numeric data.

1.1 What is Numpy?

- Numpy is an open-source add-on modules to python which provides common mathematical and numerical routines in pre-compiled, fast functions.
- The NumPy(Numeric Python) package provides basic routines for manipulating large arrays and matrices of numeric data.
- The SciPy(Scientific python) package extends the functionality of NumPy with a substantial collection of useful algorithms.

1.2 Installation and Import!!!?

- In General NumPy is pre-installed on Colab. You can check availability of NumPy and its version as:
  !pip show numpy

1.2 Installation and Import!!!?

- In General NumPy is pre-installed on Colab. You can check availability of NumPy and its version as:
  !pip show numpy
- The NumPy(Numeric Python) package provides basic routines for manipulating large arrays and matrices of numeric data.

1.2 Installation and Import!!!?

- In General NumPy is pre-installed on Colab. You can check availability of NumPy and its version as:
  !pip show numpy
- The NumPy(Numeric Python) package provides basic routines for manipulating large arrays and matrices of numeric data.
- The SciPy(Scientific python) package extends the functionality of NumPy with a substantial collection of useful algorithms.

Numpy Arrays: Introduction

- The Central feature of Numpy is the array object class. Arrays are similar to lists in Python, except that every element of an array must be of the same dtype.
- Arrays make operations with large amounts of numeric data very fast and are generally more efficient than lists.
- Example (Importing Numpy and Creating Arrays of Zeros):

```
1 import numpy as np
2 # initialize an all zero array with size 2 X 3:
3 zeros_arr = np.zeros((2,3))
4 print(" A zeros array is \n", zeros_arr, "with
    dimensions", zeros_arr.shape, "\n")
```

Task Set -I: Creating an Array.

- Based on above example.Complete the following tasks:
  1. Initialize an empty array with size $2X2$.
  2. Initialize an all one array with size $4X2$.
  3. Return a new array of given shape and type, filled with fill_value.{Hint: np.full}
  4. Return a new array of zeros with same shape and type as a given array.{Hint: np.zeros_like}
  5. Return a new array of ones with same shape and type as a given array.{Hint: np.ones_like}
  6. For an existing list `new_list = [1,2,3,4]` convert to an numpy array.{Hint: np.array()}

Array Manipulation: Numerical Ranges

- Observe the output of following code:

```
1     import numpy as np
2     range_arr = np.arange(10)
3     print("An array given range is \n", range_arr,
      "with dimensions",range_arr.shape,"\n")
4  # return evenly spaced values within a given
      interval.
5     linspace_arr = np.linspace(2.0, 3.0, num = 5,
      endpoint=false)
6     print("An evenly spaced array given range is \
      n",linspace_arr,"with dimensions",linspace_arr
      .shape,"\n")
7  # return evenly spaced numbers over a specified
      interval.
```

Task Set-II: Array Manipulation

- Numerical Ranges and Array Indexing:
  1. Create an array with values ranging from 10 to 49. {Hint:np.arrange()}.
  2. Create a $3X3$ matrix with values ranging from 0 to 8. {Hint:look for np.reshape()}
  3. Create a $3X3$ identity matrix.{Hint:np.eye()}
  4. Create a random array of size 30 and find the mean of the array. {Hint:check for np.random.random() and array.mean() function}
  5. Create a $10X10$ array with random values and find the minimum and maximum values.
  6. Create a zero array of size 10 and replace 5^{th} element with 1.
  7. Reverse an array `arr = [1,2,0,0,4,0]`.
  8. Create a 2d array with 1 on border and 0 inside.
  9. Create a $8X8$ matrix and fill it with a checkerboard pattern.

Task Set-III: Array Arithmetic

- For the following arrays:
  `x = np.array([[1,2],[3,5]])` and
  `y = np.array([[5,6],[7,8]]);`
  `v = np.array([9,10])` and `w = np.array([11,12]);`
  Perform following with numpy:
  1. Add the two array.
  2. Subtract the two array.
  3. Multiply the array with any integers of your choice.
  4. Find the square of each element of the array.
  5. Find the dot product between: `v(and)w` ; `x(and)v` ; `x(and)y`.
  6. Concatenate `x(and)y` along row and Concatenate `v(and)w` along column.
     {Hint:try `np.concatenate()` or `np.vstack()` functions.
  7. Concatenate `x(and)v`; if you get an error, observe and explain why did you get the error?

Task Set-IV: Vector Arithmetic

- Euclidean Norm of a Vector u is given by:

$$\|\mathbf{u}\| = \sqrt{u_1^2 + u_2^2 + \ldots + u_n^2}$$

- Angle Between Two Vectors u and v is given by:

$$cos\theta = \frac{u.v}{\|\mathbf{u}\|.\|\mathbf{v}\|}$$

$$\theta = arccos(\frac{u.v}{\|\mathbf{u}\|\|\mathbf{v}\|})$$

- Based on the above equation complete the code blocks in next slide:

Complete the codes

```
1     def euclidean_norm(vector):
2     "Arguments:
3     A function which calculates norm of any given
      vector.
4     inputs: vector: an 1D array
5     output: norm of a vector(a scalar)
6     "
7         import math
8         norm_vector = # write your code here
9         return norm_vector
10    import numpy as np
11    arr = [0.5, -1.2, 3.3, 4.5]
12    norm_arr = euclidean_norm(arr)
13    print("The norm of an arr \n", norm_arr "\n")
```

Complete the codes

```
1     def anglebetweenvectors(vector1, vector2):
2         "Arguments:
3         This function calculates the angle between two
      vector.
4         inputs:
5         vector1: an 1D ndarray (numpy array).
6         vector2: an 1D ndarray (numpy array).
7         outputs:
8         angle: angle between two vectors (a scalar)
9         "
10        angle = # your code here.
11        return angle
12    import numpy as np
13    arr1 = [0,1]
14    arr2 = [1,0]
15    angle = anglebetweenvectors(arr1, arr2)
16    print("The angle is\n", angle "\n")
```

2. Matrix Arithmetic with Numpy

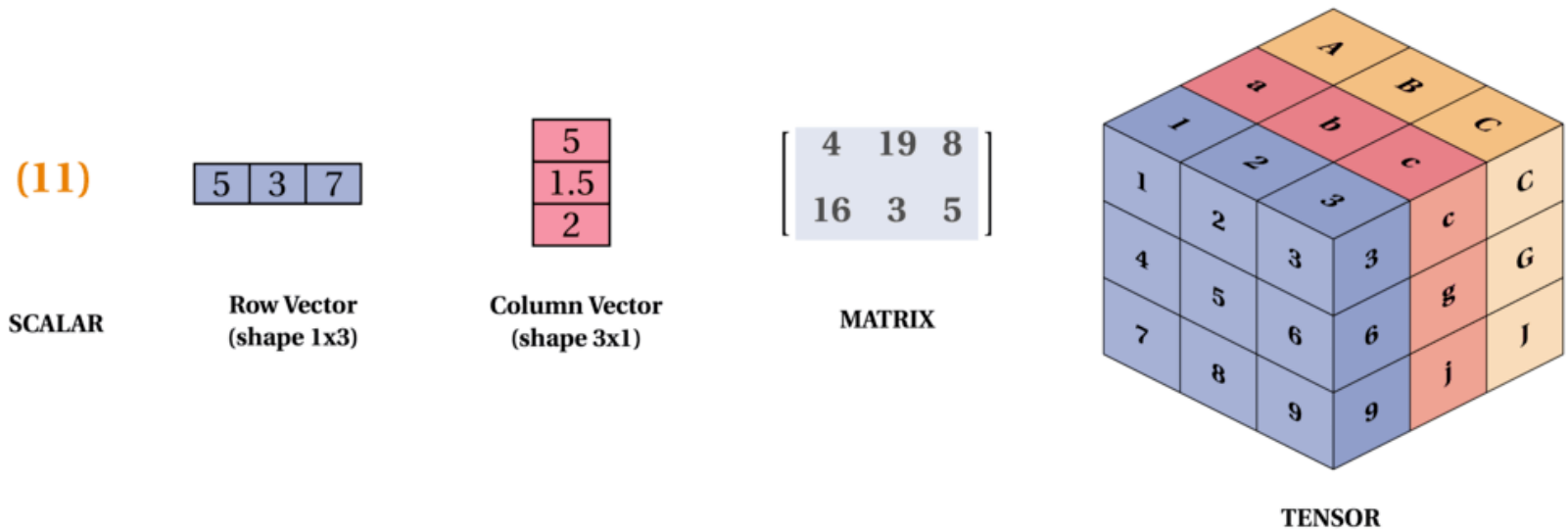

Figure: Matrix Form Representations

Matrix Additions and Multiplications

Recreate the operations shown in the image.

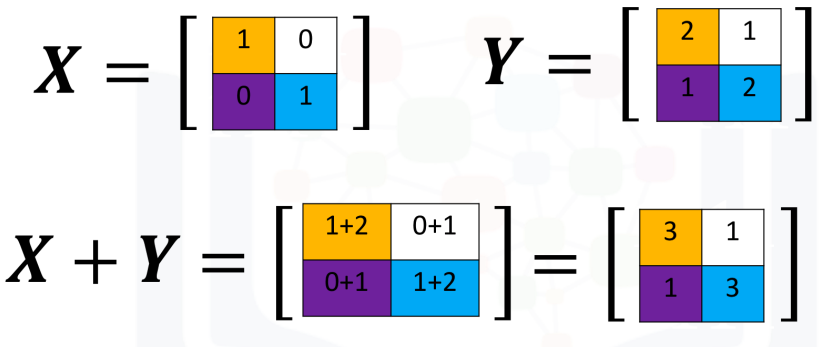

Figure: Matrix Addition

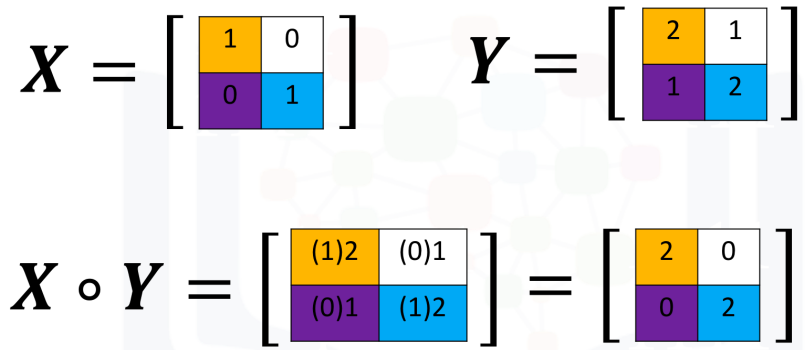

Figure: Matrix Multiplication (Perform both Hadamard and Regular Multiplication)

Task Set-V: Matrix Operations

- For the following arrays:
  `A = np.array([[3,4],[7,8]])` and
  `B = np.array([[5,3],[2,1]]);`
  Prove following with Numpy:
  1. Prove $A.A^{-1} = I$.
  2. Prove $AB \neq BA$.
  3. Prove $(AB)^T = B^T A^T$.

Task Set-VI: Solve System of Linear Equations

- Solve the following system of Linear equation using Inverse Methods.

$$\begin{aligned} 2x - 3y + z &= -1 \\ x - y + 2z &= -3 \\ 3x + y - z &= 9 \end{aligned}$$

{Hint: First use Numpy array to represent the equation in Matrix form. Then Solve for: $AX = B$}

- Now: solve the above equation using `np.linalg.inv` function.{Explore more about "linalg" function of Numpy}