Week 6

Array of Arrays

Data can be stored in 1D ,2D, 3D or *D array.

Variable	Array	Structured Array
a	[4, 5, 6, 8, 4, 9, 7, 1]	
b	[[4, 5, 6, 8, 4, 9, 7, 1]]	[[4, 5, 6, 8], [4, 9, 7, 1]]
c	[[[4, 5, 6, 8, 4, 9, 7, 1]]]	[[[4, 5], [6, 8]], [[4, 9], [7, 1]]]

Same data is stored in the all of the cases. The difference lies in the way, data is accessed.

Array	Value
a[0]	4
b[0][0]	4
c[0][0][0]	4

Matrix

In the last lecture, we saw how 1D array/list can be used to represent a vector in computer memory. A Matrix like

$$\begin{bmatrix} 2 & 1 \\ 3 & 7 \end{bmatrix}$$

can also be stored in computer memory using the same data structure (i.e. 1D list). But it can be extremely inconvenient to store a matrix in 1D array from programming point of view. I will keep on demonstrating that in the next section.

> Enjoyable Task: Storing a matrix in 1D array from programming point of view is extremely inconvenient, but think from computational point of view? Storing and solving a matrix in 1D array will consume less computer resources or storing and solving the same matrix in 2D array?

Matrix Addition Function

A Matrix can be represented in computer memory using a 2D list. Forexample,

$$\text{The vector } \begin{bmatrix} 2 & 1 \\ 3 & 7 \end{bmatrix} \text{ Translates to } [[2, 1], [3, 7]]$$

Function Arguments

Two matrices say mat1, mat2 are accepted as function arguments.

```
def add_2_matrix(mat1, mat2)
```

Validating Data

1st Check Point

`mat1` and `mat2` must be a list.

```
for mat in [mat1, mat2]:
    if isinstance(mat, list):
        pass
    else:
        raise Exception(f"{mat} must be 2D list")
```

2nd Check Point

For `mat1` and `mat2` to be a 2D array, the elements of `mat1`, `mat2` must be lists.

```
for mat in [mat1, mat2]:
    for element in mat:
        if isinstance(element, list):
            pass
        else:
            raise Exception(f"Elements of {mat} must be list.")
```

3rd Check Point

Elements of elements of `mat1` and `mat2` must be integers/floats,

```
for mat in [mat1, mat2]:
    for element in mat:
        for i in element:
            if isinstance(i, int) or isinstance(i, float):
                pass
            else:
                raise Exception(f"{i} has data type {type(i)}, whereas it
should be int/float.")
```

4th Check Point

At this point in code, we are done with computer data validity. The next thing to deal with is Math. As Matrices have some set of rules that must be followed up. 1st of them being a matrix must have equal number of elements in each row. Because for a computer

```
[[1, 2], [3]]
```

is valid but there cannot be such a matrix.

```
for mat in [mat1, mat2]:
    row_l = len(mat[0])
    for i in mat[1:]:
        if len(i) == row_l:
            pass
        else:
            raise Exception(f"length of each row must be same for a
matrix")
```

> At this point stop for the while and consider making the code for 4th check point in case the matrix is saved in a 1D array i.e. [2, 1, 3, 7]. 1st it must be checked that the matrix has even number of entries. Then considering the 1st two entries as row1 and next two entries as row2. No doubt the code will be more complex than what we have right now.

5th Check Point

After valiading the authenticity of both matrices. It must be checked if the two matrices have equal number of rows and columns.

```
if len(mat1) == len(mat2):
    pass
else:
    raise Exception(f"Both matrices must have equal number of rows")
if len(mat1[0]) == len(mat2[0]):
    pass
else:
    raise Exception(f"Both matrices must have equal number of columns")
```

> Again consider applying the algorithm on a matrix stored in 1D array.

Processing

```
return [[i + j for i,j in zip(mat1[k], mat2[k])] for k in range(len(mat1))]
```

Practice Questions

> Write a function that takes adds, subtracts, multiplies, divides an integer from a matrix (`+,-,*,/'ing Scaler with a matrix`). Function takes integer, matrix and operation (as string `"+"`, `"-"`, `"*"`, "/") as arguments and returns the result.

> Write a function that takes adds, subtracts, multiplies, divides two matrices. Function takes matrices and operation (as string `"+"`, `"-"`, `"*"`, "/") as arguments and returns the result.

Advanced Practice Questions

Matrix Multiplication Algorithm

There is one memorable formula given to perform matrix multiplication.

$$\begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \end{bmatrix} = \begin{bmatrix} a_1B_1 + a_2b_3 & a_1B_2 + a_2b_4 \\ a_3B_1 + a_4b_3 & a_3B_2 + a_4b_4 \end{bmatrix}$$

This algorithm can be programmed. But let's do matrix multiplication in a completely different manner i.e. `Matrix Multiplication` as `Linear Combination`.

$$A = \begin{bmatrix} 3 & 4 \\ 1 & 2 \end{bmatrix} and \ B = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

Multiplying:

$$AB$$

$$\begin{bmatrix} 3 & 4 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

Consider the two columns of Matrix B as vectors i.e.

$$b_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, b_2 = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

Multiply A with B_1

$$\begin{bmatrix} 3 & 4 \\ 1 & 2 \end{bmatrix} B_1$$

$$\begin{bmatrix} 3 & 4 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Think of this as a linear combination of columns of A where elements of b_1 represent values multiplication factors i.e.:

$$1 \begin{bmatrix} 3 \\ 1 \end{bmatrix} + 2 \begin{bmatrix} 4 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix} + \begin{bmatrix} 8 \\ 4 \end{bmatrix} = \begin{bmatrix} 11 \\ 5 \end{bmatrix}$$

Multiply A with B_2

$$\begin{bmatrix} 3 & 4 \\ 1 & 2 \end{bmatrix} B_2$$

$$\begin{bmatrix} 3 & 4 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

Think of this as a linear combination of columns of A where elements of b_2 represent values multiplication factors i.e.:

$$3\begin{bmatrix}3\\1\end{bmatrix}+4\begin{bmatrix}4\\2\end{bmatrix}=\begin{bmatrix}9\\3\end{bmatrix}+\begin{bmatrix}16\\8\end{bmatrix}=\begin{bmatrix}25\\11\end{bmatrix}$$

The resulting matrix (of multiplication) contains 1st column as the vector extracted by multiplying A with b_1 and 2nd column as vector extracted by multiplying A with b_2:

$$\begin{bmatrix}11 & 25\\5 & 11\end{bmatrix}$$

This process may seem tedious not only from Mathematical point of view but also from programming point of view. Why going through all of this trouble when we simply had the matrix multiplication formula. Unfortunately, this document is not supposed to get into too much Math details so I consider skipping that and try to cover the programming details. Instead of writing new code for matrix multiplication, try to reuse code (functions) from previous chapter (vectors). We have already written code/function for linear combination of vectors. Here I tried to do Matrix Multiplication manually so much that we simplify the problem to linear combination and after that just to multiply a scalar with a vector. Instead of writing new code for matrix multiplication, try to reuse some code.