matrices

August 16, 2022

0.1 Matrices

matrix is a 2 dimensional array where numbers, symbols or expressions are arranges into rows and columns. using the concept of lists, one can easily define a matrix in python.

we define 2 matrices A and B

[ ]:
```
A = [[1,2,3],[4,5,6],[7,8,9]]
B = [[9,8,7],[6,5,4],[3,2,1]]

print(A)
print(B)
```

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
[[9, 8, 7], [6, 5, 4], [3, 2, 1]]
```

[ ]:
```
# from sympy import pprint

print(A + B)
print(2*B)
```

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9], [9, 8, 7], [6, 5, 4], [3, 2, 1]]
[[9, 8, 7], [6, 5, 4], [3, 2, 1], [9, 8, 7], [6, 5, 4], [3, 2, 1]]
```

[ ]:
```
A*B #multiplication of lists containing lists isnot possible
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
c:\Users\joelv\Desktop\projects 2\math\8-06-22.ipynb Cell 5 in <cell line: 1>()
----> <a href='vscode-notebook-cell:/c%3A/Users/joelv/Desktop/projects%202/math/
 ↪8-06-22.ipynb#X10sZmlsZQ%3D%3D?line=0'>1</a> print(A*B)

TypeError: can't multiply sequence by non-int of type 'list'
```

[ ]:
```
A**2 #** or pow() doesnot work for list of lists
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
c:\Users\joelv\Desktop\projects 2\math\8-06-22.ipynb Cell 6 in <cell line: 1>()
```

```
----> <a href='vscode-notebook-cell:/c%3A/Users/joelv/Desktop/projects%202/math
↪8-06-22.ipynb#X11sZmlsZQ%3D%3D?line=0'>1</a> A**2

TypeError: unsupported operand type(s) for ** or pow(): 'list' and 'int'
```

[ ]:
```
A[0].append(10)
```

[ ]:
```
A
```

[ ]:
```
[[1, 2, 3, 10], [4, 5, 6], [7, 8, 9]]
```

the matrix A being defined as a list of lists, python supports adding of single element to rows or columns, which falsifies the very definition of a matrix

thorugh all the above command we can see that even though it is possible to define a matrix as a list of lists, matrix manipulation is not straight forward as desired

0.2 Numpy package

numpy is a library consisting of multidimensional array objects and package of functions for processing those arrays. Numpy stands for Numerical Python. The following operations can be performed in numpy

- Mathematical and logical operations on arrays
- fourier tranforms and routines for shape manipulation
- operations related to linear algebra

[ ]:
```
import numpy as np
```

numpy provides matrix() and array() functions

[ ]:
```
A1 = np.array([[1,2,3],[4,5,6],[7,8,9]])

print(A1)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

[ ]:
```
M1 = np.matrix([[1,2,3],[4,5,6],[7,8,9]])

print(M1)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

the difference between array() and matrix() ?

- array() provides n-dimensional array while matrix() is strictly 2-d

- the * and ** operator performs elementwise in array while in matrix they are used for multiplication and finding powers

common errrors

[ ]:
```
N = np.matrix([1,2,3],[4,5,6],[7,8,9])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
c:\Users\joelv\Desktop\projects 2\math\8-06-22.ipynb Cell 19 in <cell line: 1>(
----> <a href='vscode-notebook-cell:/c%3A/Users/joelv/Desktop/projects%202/math/
↪8-06-22.ipynb#X25sZmlsZQ%3D%3D?line=0'>1</a> N = np.
↪matrix([1,2,3],[4,5,6],[7,8,9])

File c:
↪\Users\joelv\AppData\Local\Programs\Python\Python310\lib\site-packages\numpy\matrixlib\defm
↪py:145, in matrix.__new__(subtype, data, dtype, copy)
    142     data = _convert_from_string(data)
    144 # now convert data to an array
--> 145 arr = N.array(data, dtype=dtype, copy=copy)
    146 ndim = arr.ndim
    147 shape = arr.shape

TypeError: Field elements must be 2- or 3-tuples, got '4'
```

0.3 Matrix manipulation

[ ]:
```
# shape function for order

M1.shape
```

[ ]: (3, 3)

[ ]:
```
print(M1.shape[0]) #no of rows
print(M1.shape[1]) #no of columns
```

```
3
3
```

[ ]:
```
# size function for number of elements
M1.size
```

[ ]: 9

[ ]:
```
# function zeroes creates an array of zeroes
np.zeros((4,4))
```

[ ]:
```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

[ ]:
```
np.ones((4,4))
```

[ ]:
```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

zeros provide an array of zeros and not a matrix

the zeroes and ones function gives float value by default np.zeros((4,4),int)

[ ]:
```
np.zeros((4,4),int)
```

[ ]:
```
array([[0, 0, 0, 0],
       [0, 0, 0, 0],
       [0, 0, 0, 0],
       [0, 0, 0, 0]])
```

[ ]:
```
np.ones((4,4),dtype = int)
```

[ ]:
```
array([[1, 1, 1, 1],
       [1, 1, 1, 1],
       [1, 1, 1, 1],
       [1, 1, 1, 1]])
```

0.4 Matrix operations

[ ]:
```
d = np.matrix([[1,2,3],[4,5,6],[7,8,9]])

e = np.matrix([[1,2,3],[4,5,6],[7,8,9]])

f = np.matrix([[1,2,3],[4,5,6],[7,8,9]])
```

[ ]:
```
d + e
```

[ ]:
```
matrix([[ 2,  4,  6],
        [ 8, 10, 12],
        [14, 16, 18]])
```

[ ]:
```
np.add(d,e)
```

[ ]:
```
matrix([[ 2,  4,  6],
        [ 8, 10, 12],
        [14, 16, 18]])
```

[ ]:
```
d - e
```

[ ]:
```
matrix([[0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]])
```

[ ]:
```
np.multiply(d,e)
```

[ ]:
```
matrix([[ 1,  4,  9],
        [16, 25, 36],
        [49, 64, 81]])
```

[ ]:
```
d.dot(f) # matrix multiplication
```

[ ]:
```
matrix([[ 30,  36,  42],
        [ 66,  81,  96],
        [102, 126, 150]])
```

[ ]:
```
e/d
```

[ ]:
```
matrix([[1., 1., 1.],
        [1., 1., 1.],
        [1., 1., 1.]])
```

[ ]:
```
np.divide(e,d)
```

[ ]:
```
matrix([[1., 1., 1.],
        [1., 1., 1.],
        [1., 1., 1.]])
```

0.4.1 Transpose of a matrix

[ ]:
```
d.transpose()
```

[ ]:
```
matrix([[1, 4, 7],
        [2, 5, 8],
        [3, 6, 9]])
```

[ ]:
```
np.transpose(d)
```

[ ]:
```
matrix([[1, 4, 7],
        [2, 5, 8],
        [3, 6, 9]])
```

[ ]:
```
d.T
```

[ ]:
```
matrix([[1, 4, 7],
        [2, 5, 8],
        [3, 6, 9]])
```

0.4.2 Upper and lower triangular matrix

[ ]:
```
ut = np.triu(d)
ut
```

[ ]:
```
array([[1, 2, 3],
       [0, 5, 6],
       [0, 0, 9]])
```

[ ]:
```
lt = np.tril(d)
lt
```

[ ]:
```
array([[1, 0, 0],
       [4, 5, 0],
       [7, 8, 9]])
```

0.5 numpy.linalg

linear algebra module

- rank, determinant, trace, inverse etc of an array
- eigen values of matrices
- matrix and vector products
- solve linear equations

0.5.1 determinant

[ ]:
```
det = np.linalg.det(d)
det
```

[ ]:
```
0.0
```

0.5.2 inverse of matrix

[ ]:
```
x = np.matrix([[1,2],[4,9]])
x**-1
```

[ ]:
```
matrix([[ 9., -2.],
        [-4.,  1.]])
```

[ ]:
```
np.linalg.inv(x)
```

```
[ ]: matrix([[ 9., -2.],
             [-4.,  1.]])
```

0.5.3 eigen values

```
[ ]: print(d)

     np.linalg.eigvals(d)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

```
[ ]: array([ 1.61168440e+01, -1.11684397e+00, -9.75918483e-16])
```

```
[ ]: h = np.matrix([[1,0,0],[0,9,0],[0,0,8]])

     np.linalg.eigvals(h)
```

```
[ ]: array([1., 9., 8.])
```

```
[ ]:
```

1 Identity matrix

verify the following properties by suitable examples - symmetric transpose of identity matrix is itself - identity matrix is nonsimgular - inverse of identity matrix is itself - eigenvalues are all 1's - multiplying any matrix by the unit matrix , gives the marix itself - We always get an identity after multiplying two inverse matrices

```
[ ]: import numpy as np
```

```
[ ]: i = np.matrix([[1,0,0],[0,1,0],[0,0,1]])

     print("identity matrix =\n ",i)

     trans = np.transpose(i)

     print("transpose of  matrix =\n ",i)

     de = np.linalg.det(i)

     print("determinant :\n ", de)

     inv = np.linalg.inv(i)
```

```
print("inverse = \n", inv)

eig = np.linalg.eigvals(i)

print("eigen values : ",eig)

a = np.matrix([[1,0,0],[0,5,0],[0,0,9]])

print("a = \n",a)

print("a*i = \n",a*i)

a_i = np.linalg.inv(a)
```

```
identity matrix =
  [[1 0 0]
 [0 1 0]
 [0 0 1]]
transpose of  matrix =
  [[1 0 0]
 [0 1 0]
 [0 0 1]]
determinant :
  1.0
inverse =
 [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
eigen values :  [1. 1. 1.]
a =
 [[1 0 0]
 [0 5 0]
 [0 0 9]]
a*i =
 [[1 0 0]
 [0 5 0]
 [0 0 9]]
```

2 Diagonal Matrix

verify the properties : - addition, multiplication of diagonal matrix gives a diagonal matrix again: - symmetric transpose of matrix is itself - determinant of diag(ai) = a1*a2… - identity matrix is nonsingular if diagonal entries are all on zero - diag(ai..)^-1 = diaag(a1$^{-1,....an}$-1) - eigen values are diagonal entries

[ ]:
```
d1 = np.matrix([[1,0,0],[0,2,0],[0,0,3]])
d2 = np.matrix([[4,0,0],[0,5,0],[0,0,6]])
```

```
print("d1 = \n",d1)
print("d2 = \n",d2)

d = d1 + d2
c = d1 * d2

print("d1 + d2 = \n",d)
print("d1 * d2 = \n", c)

tra = np.transpose(d1)
print("transpose = \n",d1)

#find inverse and eigen values of the same
```

```
d1 =
 [[1 0 0]
 [0 2 0]
 [0 0 3]]
d2 =
 [[4 0 0]
 [0 5 0]
 [0 0 6]]
d1 + d2 =
 [[5 0 0]
 [0 7 0]
 [0 0 9]]
d1 * d2 =
 [[ 4  0  0]
 [ 0 10  0]
 [ 0  0 18]]
transpose =
 [[1 0 0]
 [0 2 0]
 [0 0 3]]
```

3 Upper triangular matrix

- adding returns utm
- multiplying results in utm
- transpose will be ltm
- inverse remains utm
- determinant od diag(a1…an) = a1*a2*….an
- eigen values are diagonal entries

[ ]:
```
u1 = np.matrix([[1,4,5],[0,2,6],[0,0,3]])
u2 = np.matrix([[4,1,2],[0,5,3],[0,0,6]])
```

```python
print("u1 = \n",u1)
print("u2 = \n",u2)

u = u1 + u2
c = u1 * u2

print("u1 + u2 = \n",u)
print("u1 * u2 = \n", c)

tra = np.transpose(u1)
print("transpose = \n",u1)

de = np.linalg.det(u1)

print("determinant =\n",de)
```

```
u1 =
 [[1 4 5]
 [0 2 6]
 [0 0 3]]
u2 =
 [[4 1 2]
 [0 5 3]
 [0 0 6]]
u1 + u2 =
 [[5 5 7]
 [0 7 9]
 [0 0 9]]
u1 * u2 =
 [[ 4 21 44]
 [ 0 10 42]
 [ 0  0 18]]
transpose =
 [[1 4 5]
 [0 2 6]
 [0 0 3]]
determinant =
 6.0
```

4 Singularv Matrix

- det is 0
- a non-invertible matrix is reffered to as singlular matrix, i.e,