Week3_ClassWork

October 9, 2020

1 Week3: Classwork

- Edited by: LUXP

1.1 Question:

Generate Pascal's Triangle( )

```
            1
          1   1
        1   2   1
      1   3   3   1
    1   4   6   4   1
  1   5   10  10   5   1
1   6   15  20  15   6   1
```

[1]:

```
def triangles():
    L = [1]
    while True:
        yield L
        L = [sum(i) for i in zip([0]+L, L+[0])]

a = triangles()
# for i in range(7):
#     ss = " "
```

```
#     for d in next(a):
#         ss = ss +  str(d) + " "
#     print("{0:^20}".format(ss))

for i in range(7):
    ss = "  ".join([str(d) for d in next(a)])
    print("{0:^25}".format(ss))
```

```
            1
          1  1
         1  2  1
       1  3  3  1
      1  4  6  4  1
   1  5  10  10  5  1
 1  6  15  20  15  6  1
```

[2]:
```
def triangles(n):
    L = []
    r = [0, 1, 0]
    L.append(r[1:-1])
    for i in range(n):
        r = [x+y for x, y in zip(r[1:], r[:-1])]
        L.append(r)
        r = [0] + r + [0]
    return L

L = triangles(7)
print(L)
for li in L:
    ss = "_".join([str(s) for s in li])
    print("{0:^25}".format(ss))
```

```
[[1], [1, 1], [1, 2, 1], [1, 3, 3, 1], [1, 4, 6, 4, 1], [1, 5, 10, 10, 5, 1],
[1, 6, 15, 20, 15, 6, 1], [1, 7, 21, 35, 35, 21, 7, 1]]
            1
           1_1
          1_2_1
         1_3_3_1
        1_4_6_4_1
      1_5_10_10_5_1
    1_6_15_20_15_6_1
   1_7_21_35_35_21_7_1
```

1.2 Question:

- Find the Eigenvalues for the matrix: **M**

$$\begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}$$

Ref: eigen-paris

$$AV = VD, A = VDV^{-1}$$

- Take a try for its **Singular Value Decomposition(SVD)**

$$A = UDV^T, U^T * U = I, V^T = V^{-1}$$

Ref: [SVD](https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.svd.html)

[3]:
```
## Define the Matrix
import numpy as np

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

M
```

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

[4]:
```
## Calculate Inverse Matrix
print('Determinant:', np.linalg.det(M))

try:
    print('Inverse Matrix:\n', np.linalg.inv(M))
except:
    print('Inverse Matrix is not exist!')
```

```
Determinant: -359.9999999999997
Inverse Matrix:
 [[ 0.14722222 -0.14444444  0.06388889]
 [-0.06111111  0.02222222  0.10555556]
 [-0.01944444  0.18888889 -0.10277778]]
```

[5]:
```
## Calculate Eigenvalues
Evals, Evecs = np.linalg.eig(M)
print(Evals.shape, Evecs.shape)

Evals
```

```
(3,) (3, 3)
```

[5]: array([15.   , 4.89897949, -4.89897949])

[6]:
```
## Calculate Singular Values
u, s, vh = np.linalg.svd(M, full_matrices=True)
print(u.shape, s.shape, vh.shape)

s
```

(3, 3) (3,) (3, 3)

[6]: array([15.   , 6.92820323, 3.46410162])

[7]:
```
## M^T * M
MTM = M.T * M

MTM.shape
```

[7]: (3, 3)

[8]:
```
## Eigenvalue of M^T M
Evals, Evecs = np.linalg.eig(MTM)
print(Evals.shape, Evecs.shape)

Evals
```

(3,) (3, 3)

[8]: array([225., 12., 48.])

[9]:
```
s**2
```

[9]: array([225., 48., 12.])

[ ]:
```

```