

MA 506 Probability and Statistical Inference

Lecture 3: Scientific computations through scipy

SciPy is made to work efficiently on NumPy arrays

```
In [2]: import numpy as np
import scipy.linalg as LA
```

Linear Algebra operations

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

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

1 Multiplying two arrays/matrices

```
In [4]: A.dot(A)  # Multiplying array A by A
```

```
Out[4]: array([[ 30,  36,  42],
               [ 66,  81,  96],
               [102, 126, 150]])
```

```
In [5]: B = np.matrix(A)
B
```

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

```
In [6]: np.multiply(B,B)
```

```
Out[6]: matrix([[ 1,  4,  9],
                [16, 25, 36],
                [49, 64, 81]])
```

```
In [7]: np.matmul(B,B)
```

```
Out[7]: matrix([[ 30,  36,  42],
                [ 66,  81,  96],
                [102, 126, 150]])
```

2 Inverse of a matrix

```
In [8]: LA.inv(A)
```

```
Out[8]: array([[ 3.15251974e+15, -6.30503948e+15,  3.15251974e+15],
               [-6.30503948e+15,  1.26100790e+16, -6.30503948e+15],
               [ 3.15251974e+15, -6.30503948e+15,  3.15251974e+15]])
```

```
In [9]: LA.inv(B)
```

```
Out[9]: array([[ 3.15251974e+15, -6.30503948e+15,  3.15251974e+15],
               [-6.30503948e+15,  1.26100790e+16, -6.30503948e+15],
               [ 3.15251974e+15, -6.30503948e+15,  3.15251974e+15]])
```

3 Solving a system of equation ($Ax = b$)

```
In [10]: b = np.array([1,2,3])
         A,b
```

```
Out[10]: (array([[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]]),
         array([1, 2, 3]))
```

```
In [17]: x = LA.solve(A,b)
         x
```

```
/var/folders/d7/n2s59nxn5f38lzcmx7727n300000gn/T/ipykernel_16099/3230
241923.py:1: LinAlgWarning: Ill-conditioned matrix (rcond=2.20282e-18
): result may not be accurate.
```

```
x = LA.solve(A,b)
```

```
Out[17]: array([-0.23333333,  0.46666667,  0.1          ])
```

```
In [19]: # Ax should produce b  
A.dot(x)
```

```
Out[19]: array([1., 2., 3.])
```

4 Determinant of a matrix

```
In [20]: LA.det(A)
```

```
Out[20]: 0.0
```

5 Eigenvalue of a matrix

```
In [21]: LA.eigvals(A)
```

```
Out[21]: array([ 1.61168440e+01+0.j, -1.11684397e+00+0.j, -3.38433605e-16+0.j]  
)
```

6 Matrix decompositions

```
In [22]: U,S,V = LA.svd(A)
```

```
In [23]: U,S,V
```

```
Out[23]: (array([[ -0.21483724,  0.88723069,  0.40824829],  
                 [ -0.52058739,  0.24964395, -0.81649658],  
                 [ -0.82633754, -0.38794278,  0.40824829]]),  
          array([1.68481034e+01, 1.06836951e+00, 3.33475287e-16]),  
          array([[ -0.47967118, -0.57236779, -0.66506441],  
                 [ -0.77669099, -0.07568647,  0.62531805],  
                 [ -0.40824829,  0.81649658, -0.40824829]]))
```

```
In [24]: Q,R = LA.qr(A)  
Q,R
```

```
Out[24]: (array([[ -0.12309149,  0.90453403,  0.40824829],  
                 [ -0.49236596,  0.30151134, -0.81649658],  
                 [ -0.86164044, -0.30151134,  0.40824829]]),  
          array([[ -8.12403840e+00, -9.60113630e+00, -1.10782342e+01],  
                 [ 0.00000000e+00,  9.04534034e-01,  1.80906807e+00],  
                 [ 0.00000000e+00,  0.00000000e+00, -1.11164740e-15]]))
```

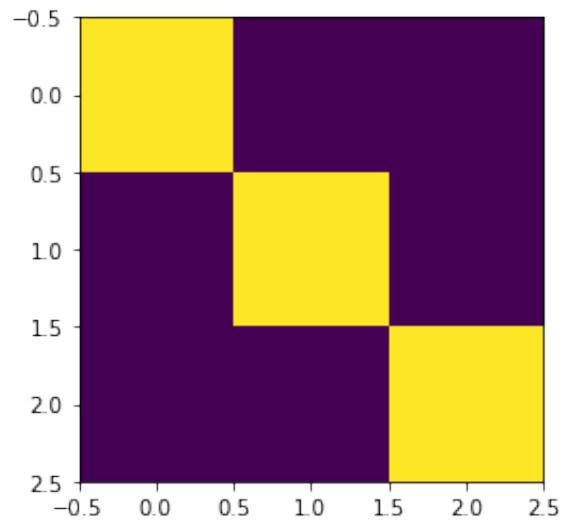
```
In [25]: Q.dot(R)
```

```
Out[25]: array([[1., 2., 3.],  
                [4., 5., 6.],  
                [7., 8., 9.]])
```

```
In [26]: Q.dot(Q.T)
```

```
Out[26]: array([[1.00000000e+00, 2.10258665e-16, 4.36991811e-16],  
                [2.10258665e-16, 1.00000000e+00, 1.43069439e-16],  
                [4.36991811e-16, 1.43069439e-16, 1.00000000e+00]])
```

```
In [27]: from matplotlib.pyplot import *  
         imshow(Q.dot(Q.T))  
         show()
```



```
In [ ]:
```