实验 6 参考解答

2022 年 10 月 11 日

1 作业一

$m \times n$ 的矩阵生成可以用 np.random.rand(m, n) 或者 np.random.randint(a, b, (m, n))，前者生成 $(0, 1)$ 之间的随机实数，后者生成 $[a, b-1]$ 内的随机正整数。注意：如果需要固定结果，应当使用 np.random.seed()。Python 自带的 random 库生成的种子对 numpy 无效。

一些我自己的小 bug：使用 scipy 的 linalg 需要用 from scipy import linalg 而不可以先 import scipy 再 scipy.linalg，可能会报错。np.linalg 无此限制。

[1]:
```
import numpy as np
from scipy import linalg
np.random.seed(114514)

A = np.random.rand(4, 4)
b = np.random.rand(4, 1)

print(A); print(b)
```

```
[[0.68775587 0.67617294 0.42661145 0.78303236]
 [0.78837491 0.03616564 0.89739725 0.75986473]
 [0.21021482 0.66824989 0.14687379 0.68313587]
 [0.39616994 0.72970845 0.04339819 0.55444743]]
[[0.18223384]
 [0.21937378]
 [0.65014185]
 [0.08114408]]
```

求解 $Ax = b$ 可以使用 numpy/scipy 内 linalg 类下的 solve() 函数。此外也可使用 $x = A^{-1}b$ 的形式。

[2]:
```
print(linalg.solve(A, b))
print(linalg.inv(A) @ b)
```

```
[[-2.17159499]
 [ 1.01154299]
 [ 1.92876951]
 [ 0.21576107]]
[[-2.17159499]
 [ 1.01154299]
 [ 1.92876951]
 [ 0.21576107]]
```

转置即 A.T，也可以用 numpy 附带的 transpose() 函数。

[3]:
```
print(A.T)
print(np.transpose(A))
```

```
[[0.68775587 0.78837491 0.21021482 0.39616994]
 [0.67617294 0.03616564 0.66824989 0.72970845]
 [0.42661145 0.89739725 0.14687379 0.04339819]
 [0.78303236 0.75986473 0.68313587 0.55444743]]
[[0.68775587 0.78837491 0.21021482 0.39616994]
 [0.67617294 0.03616564 0.66824989 0.72970845]
 [0.42661145 0.89739725 0.14687379 0.04339819]
 [0.78303236 0.75986473 0.68313587 0.55444743]]
```

行列式、逆矩阵在 numpy 或 scipy 的 linalg 中均有直接对应函数。

[4]:
```
print(linalg.det(A))
print(linalg.inv(A))
```

```
-0.00812315368612104
[[ -7.05643592   3.4964219   -3.76709738   9.81526673]
 [ 17.45883427  -7.74728573   1.21544444 -15.53665211]
 [ 22.15041683  -8.74268053   2.50156049 -22.38290554]
 [-19.66931143   8.38221786   0.89625717  16.99005141]]
```

矩阵的秩可以用 np.linalg.matrix_rank() 直接求解；**注意 scipy 的 linalg 不包含此函数**。当然也可以用特征值、奇异值等方法求解，此处略。

[5]:
```
print(np.linalg.matrix_rank(A))
```

4

特征值有两种求法：linalg.eigvals()（直接求解）或 linalg.eig()（同时求出特征值和特征向量）。对于 linalg.eig()，其返回两个变量：第一个为特征值构成的一维向量，第二个为对应的特征向量（作为列向量）构成的矩阵。

[6]:
```
# 特征值
print(linalg.eig(A)[0])
print(linalg.eigvals(A))
# 特征向量矩阵
print(linalg.eig(A)[1])
```

```
[ 2.12488287+0.j         -0.72895136+0.j          0.01465561+0.07091937j
  0.01465561-0.07091937j]
[ 2.12488287+0.j         -0.72895136+0.j          0.01465561+0.07091937j
  0.01465561-0.07091937j]
[[-0.60454524+0.j          -0.0717292 +0.j         -0.28527053-0.19205557j
  -0.28527053+0.19205557j]
 [-0.55034177+0.j           0.84599814+0.j          0.4335125 -0.12028702j
   0.4335125 +0.12028702j]
 [-0.39493206+0.j          -0.27768403+0.j          0.63923346+0.j
   0.63923346-0.j        ]
 [-0.41913918+0.j          -0.44948154+0.j         -0.46000347+0.24312696j
  -0.46000347-0.24312696j]]
```

2 作业二

[7]:
```
import numpy as np
from scipy import linalg
np.random.seed(1919810)

B = np.random.rand(6, 4)
b = np.random.rand(6, 1)
```

唯一难点是**奇异值分解**：

首先，linalg.svd() 返回的是 U，奇异值构成的一维向量 $\Lambda = (\lambda_1, \lambda_2, ...)$ 和 V^T（而不是 V），因此如果要考察 V 需要对其进行一次转置。

其次，教材里 $U = [u_1, u_2, ..., u_m]$ 中 u_i 表示的是列向量，不要搞混了。

最后，比较二者是否相等**不要用** ==，受计算机限制这里计算结果大概率存在误差。如果误差在 10^{-15} 上下浮动基本可以证明你的计算没问题。

这里先给出奇异值分解验证部分的代码：

[8]:
```
U, sigma, V = linalg.svd(B)
V = V.T
for i in range(4):
    print(B @ V[:, i] - sigma[i] * U[:, i])
```

```
[7.77156117e-16 2.22044605e-16 6.66133815e-16 1.11022302e-16
 2.22044605e-16 4.44089210e-16]
[ 0.00000000e+00  1.11022302e-16 -3.05311332e-16 -3.33066907e-16
 -5.55111512e-17 -2.22044605e-16]
[ 1.17961196e-16 -1.38777878e-16 -6.93889390e-17 -2.77555756e-17
  0.00000000e+00 -3.88578059e-16]
[ 0.00000000e+00 -3.46944695e-17  1.11022302e-16 -1.38777878e-17
  0.00000000e+00 -1.11022302e-16]
```

此外，svd() 得到的奇异值一维向量如果要还原为 $a \times b$ 的矩阵，可以使用 linalg.diagsvd(M, a, b)。此处不作阐述。

LU 分解和最小二乘直接照抄函数即可。LU 分解函数得到的三个矩阵恰好为 P, L, U。

[9]:
```
# 最小二乘法
print(linalg.inv(B.T @ B) @ B.T @ b)

# LU 分解
print(linalg.lu(B))
```

```
[[ 0.48898968]
 [ 0.47185497]
 [ 0.59605181]
 [-0.71213407]]
(array([[0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 0., 1.],
       [0., 1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0., 0.]]), array([[ 1.        ,  0.        ,  0.
```

```
,  0.        ],
       [ 0.49526724,  1.        ,  0.        ,  0.        ],
       [ 0.3566333 ,  0.99995977,  1.        ,  0.        ],
       [ 0.71223074, -0.15195158,  0.50060819,  1.        ],
       [ 0.49448553,  0.05572959,  0.6368847 ,  0.4382395 ],
       [ 0.86036441,  0.83058448,  0.41045453,  0.61298306]]), array([[
0.95920327,  0.28958468,  0.59518342,  0.87581511],
       [ 0.        ,  0.75113915, -0.28132467, -0.13127975],
       [ 0.        ,  0.        ,  0.74061948,  0.54098326],
       [ 0.        ,  0.        ,  0.        , -0.72488194]]))
```

补充：linalg 中有解超定方程组的最小二乘函数 linalg.lstsq(A, b)，可以用于验证你求得解的正确性。

[10]:
```
print(linalg.lstsq(B, b))
```

```
(array([[ 0.48898968],
       [ 0.47185497],
       [ 0.59605181],
       [-0.71213407]]), array([0.07023595]), 4, array([2.79095447, 0.99202424,
0.50480669, 0.483764  ]))
```