数学基础 6th

1. $A^TAx=A^Tb$ $\qquad A^TA=\begin{pmatrix}1&3&5\\2&4&6\end{pmatrix}\begin{pmatrix}1&2\\3&4\\5&6\end{pmatrix}=\begin{pmatrix}35&44\\44&56\end{pmatrix}$

$$\underset{2\times2}{\begin{pmatrix}35&44\\44&56\end{pmatrix}}\underset{2\times1}{x}=\begin{pmatrix}1&3&5\\2&4&6\end{pmatrix}\begin{pmatrix}1\\1\\1\end{pmatrix}=\underset{2\times1}{\begin{pmatrix}9\\12\end{pmatrix}}$$

$$x=\begin{pmatrix}-1\\1\end{pmatrix}$$

2.

$$A^TA=\begin{pmatrix}1&2&1\\3&0&0\\1&0&0\\1&0&0\end{pmatrix}\begin{pmatrix}1&3&1&1\\2&0&0&0\\1&0&0&0\end{pmatrix}=\begin{pmatrix}6&3&1&1\\3&9&3&3\\1&3&1&1\\1&3&1&1\end{pmatrix}$$

$$A^Tb=\begin{pmatrix}1&2&1\\3&0&0\\1&0&0\\1&0&0\end{pmatrix}\begin{pmatrix}1\\1\\1\end{pmatrix}=\begin{pmatrix}4\\3\\1\\1\end{pmatrix}$$

$$\begin{pmatrix}6&3&1&1\\3&9&3&3\\1&3&1&1\\1&3&1&1\end{pmatrix}x=\begin{pmatrix}4\\3\\1\\1\end{pmatrix}$$

$$\begin{pmatrix}6&3&1&1&4\\3&9&3&3&3\\1&3&1&1&1\\1&3&1&1&1\end{pmatrix}\to\begin{pmatrix}6&3&1&1&4\\1&3&1&1&1\\0&0&0&0&0\\0&0&0&0&0\end{pmatrix}\to\begin{pmatrix}6&3&1&1&4\\0&15&5&5&2\\0&0&0&0&0\\0&0&0&0&0\end{pmatrix}$$

$x_1=(4-3x_2-x_3-x_4)/6$

$x_2=(2-5x_3-5x_4)/15$

$$\begin{pmatrix}\frac{3}{5}\\\frac{2}{15}\\0\\0\end{pmatrix}+c_1\begin{pmatrix}\frac{3}{5}\\-\frac{1}{5}\\1\\0\end{pmatrix}+c_2\begin{pmatrix}\frac{3}{5}\\-\frac{1}{5}\\0\\1\end{pmatrix}\qquad\begin{pmatrix}x_1\\x_2\\x_3\\x_4\end{pmatrix}=\begin{pmatrix}\frac{3}{5}\\\frac{12}{15}\\0\\0\end{pmatrix}+c_1\begin{pmatrix}\frac{3}{5}\\-\frac{1}{5}\\1\\0\end{pmatrix}+c_2\begin{pmatrix}\frac{3}{5}\\-\frac{1}{5}\\0\\1\end{pmatrix}$$

$(c_1,c_2\in\mathbb{R})$

3、 $A\in R^{m\times n}$ $X\in R^{n\times m}$ $b\in R^m$ $x=\underset{n\times m}{X}\underset{m\times 1}{b}\in R^n$

①

Ax $m\times 1$ b $m\times 1$

$Ax-b$ $m\times 1$

$\|Ax-b\|_2$ 是 $m\times 1$ 向量的模 $\Rightarrow \|AXb-b\|_2$

极小化即 $AXb=b$

$A=(a_1,a_2,\cdots a_n)$ $a_i\in R^{m\times 1}$ 取 $b_1=a_1$ $x=\begin{pmatrix}1\\0\\0\\0\\0\\\vdots\\0\end{pmatrix}$ $n\times 1$

$\begin{pmatrix}1\\0\\\vdots\\0\end{pmatrix}=Xb_1$ $b_1=a_1$

$\underline{AX}b_1=AXa_1=a_1$

以此类推

令 b 依次为 $a_1,a_2\cdots a_n$，总存在 $X\in R^{n\times m}$，s.t.

对每一个 b_i 都有 $x=\begin{pmatrix}0\\\vdots\\1\\\vdots\\0\\0\end{pmatrix}=Xb_i$ 第 i 个分量为 1，其余为 0

$\therefore \underset{m\times m}{AX}\underset{m\times 1}{b_i}=a_i=b_i$

$A\begin{pmatrix}1\\0\\\vdots\\0\end{pmatrix}=a_1$ $AX(b_1,b_2,\cdots b_n)=(b_1,b_2\cdots b_n)$

$A\begin{pmatrix}0\\1\\0\\\vdots\\0\end{pmatrix}=a_2$ $AXA=A$，得证

$A\begin{pmatrix}0\\0\\1\\0\\\vdots\\0\end{pmatrix}=a_3$

②

$x=Xb$ 均极小化 $\|Ax-b\|_2$

$\therefore x=Xb$ 为方程 $Ax=b$ 的解

$A^TAXb=A^Tb$ $\because b$ 有任意性

$\therefore A^TAX=A^T$

左右同乘 X^T

$X^TA^TAX=X^TA^T$

$(AX)^TAX=(AX)^T$

$((AX)^TAX)^T=((AX)^T)^T$

$(AX)^T(AX)=AX=(AX)^T$ 得证

4、

$$\|A(x+\alpha w)-b\|_2^2 = \|Ax-b\|_2^2 + 2\alpha w^T A^T(Ax-b) + \alpha^2\|Aw\|_2^2$$

∵ $x \in X_{LS}$

∴ x可以使 $\|Ax-b\|_2$ 最小化

∴ 当 $\alpha=0$ 时 $\|A(x+\alpha w)-b\|_2^2$ 最小

令 $\|A(x+\alpha w)-b\|_2^2 = F(\alpha) \Rightarrow F(0)=\min F(\alpha)$

此时 $$\alpha = \frac{-\cancel{2}w^TA^T(Ax-b)}{\cancel{2}\|Aw\|_2^2} = 0$$

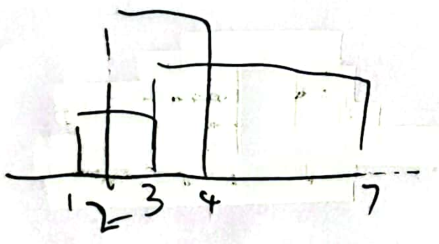
∴ $w^TA^T(Ax-b)=0$

∴ $A^TAx = A^Tb$

5、 $D_1: |\lambda-5| \le 2$ $D_2: |\lambda-2| \le 1$ $D_3: |\lambda-3| \le 1$

$A^T=A$ ∴ $AA^T=A^TA$ ∴ A为正规阵

条件数为 $\frac{|\lambda_{max}A|}{|\lambda_{min}A|} = \frac{|\lambda_1|}{|\lambda_3|}$



$-2 \le \lambda-5 \le 2$ $-1 \le \lambda-2 \le 1$ $-1 \le \lambda-3 \le 1$

$3 \le \lambda \le 7$ $1 \le \lambda \le 3$ $2 \le \lambda \le 4$

∴ 三个特征值在 $[1,7]$ 之间

∴ $\max\left(\frac{|\lambda_1|}{|\lambda_3|}\right) = 7$

∴ $\frac{|\lambda_1|}{|\lambda_3|} \le 7$

代码运行结果 $|\lambda_{max}| = |\lambda_1| = 5.65$

$|\lambda_{min}| = |\lambda_3| = 1.623$ ∴ $\frac{|\lambda_1|}{|\lambda_3|} = \frac{5.651}{1.623} = 3.482$

```python
import numpy as np


def eig_power(A, v0, pre):
    u = v0
    flag = 1
    val_old = 0
    n = 0
    while flag:
        n = n + 1
        vk = A * u
        val = vk[np.argmax(np.abs(vk))]
        u = vk / val
        if (np.abs(val - val_old) < pre):
            flag = 0
        val_old = val
        print(np.asarray(u).flatten(), val)
    print('max eigenvalue:', val)
    print('eigenvector:', np.asarray(u).flatten())
    print('iteration:', n)
    return val, u


if __name__ == '__main__':
    A = np.matrix([[5, -1, 1],
                   [-1, 2, 0],
                   [1, 0, 3]], dtype='float')
    v0 = np.matrix([[1], [1], [1]], dtype='float')
    pre = 1e-8
```

```python
    val, u = eig_power(A, v0, pre)
```

```
max eigenvalue: [[5.65109341]]
eigenvector: [ 1.         -0.27389055  0.37720286]
iteration: 27
```

```python
import numpy as np


def eig_invpower(A, v0, pre, p=0):
    uk = v0
    flag = 1
    val_old = 0
    n = 0
    if p != 0:
        A = A - p * np.eye(len(A))
    LU, La, Ua, order0, order1 = Doolittle_pivot(np.asarray(A))  # PA=LU
    while flag:
        n = n + 1
        vk = solveLineq(La, Ua, np.asarray(uk)[order1, :])
        val = vk[np.argmax(np.abs(vk))]
        uk = np.asmatrix(vk.reshape(len(A), 1)) / val
        print(np.asarray(uk).flatten())
        if (np.abs(1 / val - val_old) < pre):
            flag = 0
        val_old = 1 / val
    print('min eigenvalue:', 1 / val + p)
    print('eigenvector:', np.asarray(uk).flatten())
    print('iteration:', n)
    return 1 / val + p, uk


def Doolittle_pivot(A):  # A为np.array，而不是np.matrix
    n = len(A)
    LU = A.copy()
```

```python
    LU = A.copy()
    order1 = np.arange(n)
    for r in range(n):
        ut = LU[:r, r].reshape(r, 1)
        si = A[r:, r] - np.sum(ut * LU[r:, :r].T, axis=0)
        ir = np.argmax(np.abs(si))
        if ir != 0:
            LU[[r, r + ir], :] = LU[[r + ir, r], :]
            order1[[r, r + ir]] = order1[[r + ir, r]]
        lt = LU[r, :r].reshape(r, 1)
        LU[r, r:] = LU[r, r:] - np.sum(lt * LU[:r, r:], axis=0)
        if r == n - 1:
            continue
        LU[r + 1:, r] = (LU[r + 1:, r] - np.sum(ut * LU[r + 1:, :r].T, axis=0)) / LU[r, r]
    U = np.triu(LU)
    L = np.tril(LU) - np.diag(np.diag(LU)) + np.eye(n)
    order0 = []
    [order0.insert(i, np.where(order1 == i)[0][0]) for i in range(n)]
```

```python
        LU[r + 1:, r] = (LU[r + 1:, r] - np.sum(ut * LU[r + 1:, :r].T, axis=0)) / LU[r, r]
    U = np.triu(LU)
    L = np.tril(LU) - np.diag(np.diag(LU)) + np.eye(n)
    order0 = []
    [order0.insert(i, np.where(order1 == i)[0][0]) for i in range(n)]
    return LU, L, U, order0, order1


def solveLineq(L, U, b):  # b为np.array，而不是np.matrix
    rows = len(b)
    y = np.zeros(rows)
    y[0] = b[0] / L[0, 0]
    for k in range(1, rows):
        y[k] = (b[k] - np.sum(L[k, :k] * y[:k])) / L[k, k]
    x = np.zeros(rows)
    k = rows - 1
    x[k] = y[k] / U[k, k]
    for k in range(rows - 2, -1, -1):
        x[k] = (y[k] - np.sum(x[k + 1:] * U[k, k + 1:])) / U[k, k]
    return x


if __name__ == '__main__':
    A = np.matrix([[5, -1, 1],
                   [-1, 2, 0],
                   [1, 0, 3]], dtype='float')
    v0 = np.matrix([[1], [1], [1]], dtype='float')
    pre = 1e-10
    val, uk = eig_invpower(A, v0, pre, 1.2679)
```

```
min eigenvalue: 1.6227971460423911
eigenvector: [ 0.37720285  1.          -0.27389055]
iteration: 17
```