

[〆切] 2020/12/4 19:00

[VIII. 最小 2 乗法]

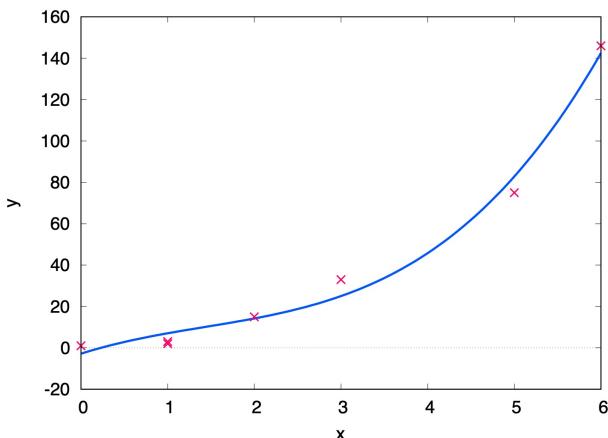
VIII-B. (1) VIII-A の問 3 にある 8 つのデータ点を最小 2 乗法を用いて 3 次多項式 $p_3(x)$ を当てはめる問題を考える。正規方程式を LU 分解で解くプログラムを作成し $p_3(x)$ の係数を有効数字 4 衔で 5 衔目を四捨五入して求めよ。作成したプログラムも提出すること。プログラミング言語は問わない。

(2) 問 1 の正規方程式をハウスホルダー法を用いて QR 分解してから解くプログラムを作成し $p_3(x)$ の係数を有効数字 4 衔で 5 衔目を四捨五入して求めよ。作成したプログラムも提出すること。プログラミング言語は問わない。

(1) V-A 等で作成した LU 分解のプログラムにおいて $\begin{cases} A \rightarrow X^T X \\ L \rightarrow X^T y \end{cases}$ を変更して解けばよい。

$$P_3(x) = a + b x + c x^2 + d x^3 \text{ としておき。}$$

$$a = -2.827, \quad b = 13.33, \quad c = -4.514, \quad d = 1.055 \text{ とする。}$$



残差の最小値 195.0

“あり” $P_3(x) = a_0 + a_1 x$ を用いた場合の 2112.54 である。

キーワード

ハウスホルダー-方法

(2) 数値は問 1 と同じ

```

for j in range(n):
    norm_x = 0
    for i in range(j, m):
        norm_x = norm_x + X[i][j]**2
    norm_x = np.sqrt(norm_x)
    if X[j][j] >= 0:
        d = norm_x
    else:
        d = -norm_x
    g = np.sqrt(norm_x*(norm_x + abs(X[j][j])))

    U[j][j] = (X[j][j] + d) / g
    X[j][j] = -d
    for i in range(j+1, m):
        U[i][j] = X[i][j] / g
        X[i][j] = 0

    temp = np.zeros(m*n).reshape(m, n)
    for i in range(j, m):
        for k in range(j+1, n):
            sum = 0
            for l in range(j, m):
                sum = sum + U[l][j] * X[l][k]
            temp[i][k] = X[i][k] - sum * U[i][j]
    for i in range(j, m):
        for k in range(j+1, n):
            X[i][k] = temp[i][k]

```

```

for i in range(m):
    H[i][i] = 1

temp = np.zeros(m*m).reshape(m, m)
for l in range(n):
    for i in range(m):
        for j in range(m):
            sum = 0
            for k in range(m):
                if i == k:
                    sum = sum + (1 - U[i][l] * U[k][l]) * H[k][j]
                else:
                    sum = sum + (-U[i][l] * U[k][l]) * H[k][j]
            temp[i][j] = sum
    for i in range(m):
        for j in range(m):
            H[i][j] = temp[i][j]

for j in range(n):
    sum = 0
    for k in range(m):
        sum = sum + H[j][k] * y[k]
    z[j] = sum

alpha = np.zeros(n)

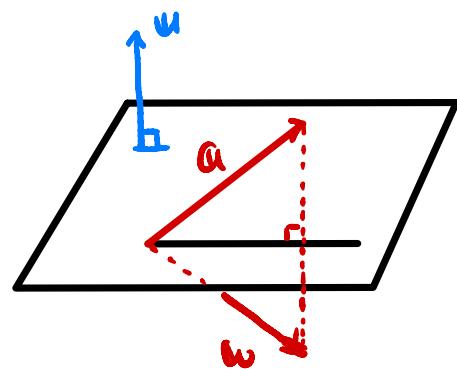
# Backward substitution
alpha[n-1] = z[n-1] / X[n-1][n-1]
for i in range(n-2, -1, -1): # n-2, n-1, ..., 0
    sum = 0
    for j in range(i+1, n): # i+1, i+2, ..., n-1
        sum = sum + X[i][j]*alpha[j]
    alpha[i] = (z[i] - sum) / X[i][i]

```

④ Householder 变換による直交化

方針 鏡像変換行列 $H(u) = I - uu^T$ ($\|u\| = \sqrt{2}$)
(直交行列かつ対称行列 $H(u)^T = H(u)$)

を使つて行列の中にある列ベクトルを修正し
余分な要素を0にする。



確認 • 対称行列？

$$H(u)^T = I^T - (uu^T)^T = I - uu^T = H(u) \quad \text{Yes!}$$

• 直交行列？

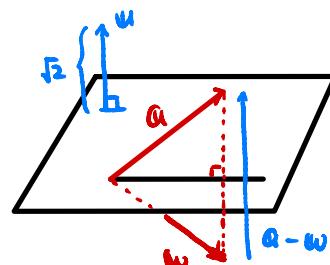
$$H(u)^T H(u) = H(u) H(u)^T = H(u)^2 = (I - uu^T)^2$$

$$= I - 2uu^T + \underbrace{uu^Tuu^T}_{\stackrel{T}{\rightarrow} \|u\|_2^2 = 2} = I \quad \text{Yes!}$$

定理 a と w が $w = H(u)a$ ($a = H(u)w$) の関係に
ある時、 u は

$$u = \sqrt{2} \frac{a - w}{\|a - w\|} \quad \dots \textcircled{1}$$

とする。



これを図で理解

方針を思い出す

$H(u)$ を使って行列中の列ベクトル $a = [a_1, a_2, \dots, a_n]$ の要素を
必要なものを残して0にしていく。

→ w にて要素を1つしか持たないものを考え。 ($w = \begin{bmatrix} -d \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = -d e_1$)

解説

変換してベクトル a と変換後に落ちついてほしい
ベクトル w の形を指定した後に平面を決める u を選ぶ。

$$\|\omega\| = \|d\| = \|\alpha\| \quad (\text{鏡像変換しても長さは変わらない})$$

d の符号は $\alpha - \omega$ の行落ちが生じないよう $\text{sign}(\alpha_i) \times c_3$

つまり

$$\begin{aligned}\alpha - \omega &= \alpha - (-d e_i) = \alpha + d e_i \\ &= \alpha + (\text{sign}(\alpha_i) \|\alpha\|) e_i\end{aligned}$$

$$\begin{aligned}\|\alpha - \omega\| &= 2 \|\alpha\|^2 + 2 \text{sign}(\alpha_i) \alpha_i \|\alpha\| \\ &= 2 \|\alpha\| \left\{ \|\alpha\| + |\alpha_i| \right\}\end{aligned}$$

$$H(u) \in S^1 \subset \mathbb{R}^n \quad H(u)y = (I - uu^\top)y = y - u(u^\top y)$$

↑ これは u を x で除算した y を回転させた。

$$\begin{aligned}H(u_i)A &= [H(u_i)a_1, H(u_i)a_2, \dots, H(u_i)a_n] \\ &= [\omega_i, H(u_i)a_2, \dots, H(u_i)a_n]\end{aligned}$$

$$\frac{\sqrt{2}(\alpha_i + \sigma)}{\|A - \alpha I\|_2} \times L(\sigma)$$

$$w_i = \underbrace{\text{sign}(a_i) \|a_i\| e_i}_{\sigma} = -\sigma e_i$$

$$(X^T X) \alpha = X^T y \dots \textcircled{1}$$

$$\left(\alpha = (X^T X)^{-1} X^T y \dots \textcircled{2} \right)$$

$$X = QR$$

$\begin{matrix} n \times n & m \times n \\ m \times n & n \times n \\ n \times n & n \end{matrix}$

 $\begin{bmatrix} * & * & \cdots & * \\ * & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$

$$Q^T Q = I_{n \times n}, (Q Q^T \neq I_{m \times m})$$

$$X^T X = R^T Q^T Q R = R^T R \dots \textcircled{3}$$

$$(\textcircled{1} \text{ の左} \textcircled{2}) = R^T R \alpha$$

$$(\textcircled{1} \text{ の右} \textcircled{2}) = R^T Q^T y$$

$$\therefore R^T R \alpha = R^T Q^T y$$

$R \alpha = Q^T y$ ← 既約式
上: A ← 逆元計算
↓ 順代入法

$$U = H^{(n)} H^{(n-1)} \dots H^{(1)}$$

$$= (I - u_n u_n^T) (I - \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & u_{n-1} u_{n-1}^T \end{bmatrix}}_{m \times m}) \dots (\underbrace{(I - u_1 u_1^T)}_{m \times m})$$

$$= [Q, Q_1]$$

$$\begin{matrix} \uparrow R^{m \times n} \\ R^{m \times (m-n)} \end{matrix}$$

[〆切] 2020/12/4 19:00

[IX. 固有値問題と特異値分解]

IX-A. VIII-A の問 3 にある 8 つのデータ点を用いて 2×2 の分散・共分散行列を構成し、ベキ乗法を用いて最大固有値とそれに対応する固有ベクトルを求めるプログラムを作成せよ。また分散・共分散行列が対称であり固有ベクトル同士が互いに直交する性質を利用してもう一つの固有値と固有ベクトルを求めよ。固有ベクトルは長さを 1 に正規化し、値は有効数字 4 桁で 5 桁目を四捨五入して求めよ。

※ 固有ベクトルを決める際に残る向きの自由度はどちらの方向を選んでも構わない。また分散共分散行列を求める際に不变分散を用いても良い。

最大固有値 2573

対応する固有ベクトル [0.03835, 0.9993]

もう一つの固有値 0.5025

対応する固有ベクトル [-0.9993, 0.03835]

キーワード
ベキ乗法

```
for k in range(10):
    z = u/np.linalg.norm(u)
    u = A @ z
    l = np.dot(z, u)
    print(k, z, u, l)
    if np.linalg.norm(u - l * z) <= 10**-15:
        print(k, z, u, l)
        break
print(z, u, l)
```

w = [-z[1], z[0]]

