Hinatori8

2026 年 1 月 11 日

2. QR Factorization and Least Squares

2.1 Projectors

Projectors

$v \in \text{range}(P)$ の場合、v は自分自身の影の上にあることになる。Mathematically,... は、$v \in \text{range}(P)(v = Px)$ であるならば、もう一度射影しても何も変わらない $(Pv = P^2x = v)$

Complementary Projectors

$$(I - P)^2 = I - P$$

は projector Pが満たす、$P^2 = P$を満たすため、$I - P$も射影である。

Given v, find vectors $v_1 \in S_1$ and $v_2 \in S_2$ such that $v_1 + v_2 = v$ で、

$$(Pv + v_3) + ((I - P)v - v_3) = v$$

を考える。$v_1 + v_2 = v$ との比較より、

$$Pv + v_3 \in S_1, \qquad (I - P)v - v_3 \in S_2$$

を満たす。つまり、v を分解したベクトル v_1, v_2 から v_3 だけズレた場合を考えている。しかし、

$$\text{range}(P) = S_1,\ \text{null}(P) = S_2 \implies \text{range(P)} \cap \text{null}(P) = \{0\}$$

であるから、$v_3 = 0$ に違いない。これは、

$$v = v_1 + v_2$$

は一意に定まるという意味である。

射影行列 Pが直交射影であるならが、$P = P^*$ となる。この時、

$$\text{range}(P) \perp \text{range}(I - P)\ (= \text{null}(P))$$

つまり、直交直和に分解ができるため、

$$\mathbb{F}^n = \text{range}(P) \oplus \text{null}(P)$$

でもある。

Projectors with an Orthonormal Basis

orthogonal Projectors に対して、Full で特異値分解を行うと、

$$\Sigma = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & 0 & \\ & & & & \ddots \end{bmatrix} \quad (= Q^* P Q)$$

となるので、Reduced で行うと、

$$P = \hat{Q}\hat{Q}^* \quad (\Sigma = I)$$

C^m の正規直交基底 q_j を列ベクトルに持つ行列 $\hat{Q}$ である。この $\hat{Q}$ は SVD を用いなくても自然に現れる。直交射影 P は

1. $P^2 = P$
2. $P^* = P$

を満たす。この $\hat{Q}\hat{Q}^*$ はそれを満たすため、直交射影である。
$\hat{Q}$ の列が正規直交であるならば、$\hat{Q}\hat{Q}^*$ は必ず range($\hat{Q}$) への直交射影を行う。
これを示したい。

■ **方針**　次の 2 つを示せれば良い

1. $\hat{Q}\hat{Q}^* v \in \text{range}(\hat{Q})$
2. $y^*(v - \hat{Q}\hat{Q}^* v) = 0$ for $y \in \text{range}(\hat{Q})$

1.

$$\hat{Q}\hat{Q}^* = \begin{bmatrix} | & | & & | \\ q_1 & q_2 & \cdots & q_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} - & q_1^* & - \\ - & q_2^* & - \\ & \vdots & \\ - & q_n^* & - \end{bmatrix} = \sum_{i=1}^{n} (q_i q_i^*)$$

任意の $v \in \mathbb{C}^m$ に対し、

$$\hat{Q}\hat{Q}^* v = \sum_{i=1}^{n} (q_i q_i^*) v = \sum_{i=1}^{n} q_i (q_i^* v)$$

つまり、$\hat{Q}\hat{Q}^* v$ 各列に対して $q^* v$ 倍したものと言え、依然として q_i を基底ベクトルとしている。

$$\hat{Q}\hat{Q}^* v \in \text{range}(\hat{Q})$$

が示せた。

2.

任意に $y \in \text{range}(\hat{Q})$ を取ると、ある $c \in \mathbb{C}^r$ が存在して、$y = \hat{Q}c$ とかける。この時、

$$\begin{aligned} y^*(v - \hat{Q}\hat{Q}^*v) &= (Qc)^*(v - \hat{Q}\hat{Q}^*v) \\ &= c^*(\hat{Q}^*v - \hat{Q}^*\hat{Q}\hat{Q}^*v) \\ &= c^*(\hat{Q}^*v - \hat{Q}^*v) \\ &= 0 \end{aligned}$$

つまり、

$$v - \hat{Q}\hat{Q}^* \perp \text{range}(Q)$$

以上より、$\hat{Q}\hat{Q}^*$ は $\text{range}(\hat{Q})$ への直交射影である。

q は C^m の正規直交基底であったが、$\text{range}(A)$ の正規直交基底を列ベクトルとする行列が Q と考えれば、QQ^* は $\text{range}(A)$ への直交射影と言える。$\|a\| = 1$ ならば、$(aa^*)a = a$ となるため、aa^* は $\text{span}\{a\}$ への直交射影となる。ならば、

$$P = \frac{aa^*}{a^*a}$$

は直交射影となる。

2.2 QR Factorization

Reduced and Full QR Factorizations

Full QR Factorization では行列 R の下側の行は 0 となるが、その行にかけられる $m - n$ 列の q は 0 であっていいわけではない。Reduced QR Factorization での q は行列 A の列空間を張る基底となっているが、0 がかけられる q は range(A) の直交補空間 $\text{range}(A)^\perp$ を張る基底となる。つまり、Full QR Factorization での q は $\mathbb{R}^m$ 全体の基底を張ることになる。

$$\text{range}(A)^\perp = \{x \in \mathbb{R}^m \mid x^Ty = 0,\ \forall y \in \text{range}(A)\}$$

$$\mathbb{R}^m = \text{range}(A) \oplus \text{range}(A)^\perp$$

Gram-Schmidt Orthogonalization

式 (7.5) について

$$r = x - (q_1^*x)q_1 - \cdots - (q_n^*x)q_n$$

を参考にして作成されている。任意のベクトル x に対して、表現可能であった。では、$q_1, \ldots, q_{j-1}$ に対して直交するベクトル v_j を考えるとなると、

$$v_j = x - (q_1^*x)q_1 - \cdots - (q_{j-1}^*x)q_{j-1}$$

となる。q_j を含むと、$r = v_j = 0$ となってしまう。x は任意であるなら、a_j を用いれば、a_j を v_j に変換する式とみることが可能となる。

$$\|x\|_2 = \sqrt{x^* x}$$

より、

$$\begin{aligned} |r_{jj}| = \sqrt{r_{jj}^* r_{jj}} &= \|r_{jj}\|_2 \\ &= \left\| \frac{a_j - \sum_{i=1}^{j-1} r_{ij} q_i}{q_j} \right\|_2 \quad \text{(式 (7.6) より)} \\ &= \left\| a_j - \sum_{i=1}^{j-1} r_{ij} q_i \right\|_2 \end{aligned}$$

この時、q は式 (7.6) より正規化しているので、$\|q_j\|_2 = 1$ を満たす。

When Vectors Become Continuous Functions

$$A = \left[\begin{array}{c|c|c|c|c} & & & & \\ 1 & x & x^2 & \cdots & x^{n-1} \\ & & & & \end{array}\right] = \left[\begin{array}{c|c|c|c} & & & \\ q_0(x) & q_1(x) & \cdots & q_{n-1}(x) \\ & & & \end{array}\right] \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & & \vdots \\ & & \ddots & \\ & & & r_{nn} \end{bmatrix}$$

より、

$$x^k = \sum_{j=0}^{k} r_{jk} q_j(x)$$

q_j は次数 j の多項式となる。これらの多項式は **Legendre polynomials**(ルジャンドル多項式)P_j のスカラー倍となる。P は互いに直交しているので、数値計算で利点がある。ルジャンドル多項式は行列 A(a continuous analogue of the Vandermonde matrix) に対して QR 分解した際に、導かれた物である。教科書では $P_j(1) = 1$ になるように正規化している。

例えば、3 次で検証すると、

$$A = QR = \begin{bmatrix} q_0(x) & q_1(x) & q_2(x) \end{bmatrix} \begin{bmatrix} \langle q_0, 1\rangle & \langle q_0, x\rangle & \langle q_0, x^2\rangle \\ 0 & \langle q_1, x\rangle & \langle q_1, x^2\rangle \\ 0 & 0 & \langle q_2, x^2\rangle \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & \sqrt{\frac{3}{2}}x & \sqrt{\frac{45}{8}}\left(x^2 - \frac{1}{3}\right) \end{bmatrix} \begin{bmatrix} \sqrt{2} & 0 & \frac{\sqrt{2}}{3} \\ 0 & \sqrt{\frac{2}{3}} & 0 \\ 0 & 0 & \sqrt{\frac{8}{45}} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}}P_0(x) & \sqrt{\frac{3}{2}}P_1(x) & \frac{2}{3}\sqrt{\frac{45}{8}}P_2(x) \end{bmatrix} \begin{bmatrix} \sqrt{2} & 0 & \frac{\sqrt{2}}{3} \\ 0 & \sqrt{\frac{2}{3}} & 0 \\ 0 & 0 & \sqrt{\frac{8}{45}} \end{bmatrix}$$

で確かに、P_j のスカラー倍となっていることがわかる。

Solution of $Ax = b$ by QR Factorization

$Rx = Q^*b$ で前進消去を終えた形と同じになる。しかし、QR 分解は標準的な手法ではなく、ガウスの消去法の方が実務では使われる。

2.3 Gram-Schmidt Orthogonalization

Gram–Schmidt Projections

P_j は $\mathrm{span}\{q_1, q_2, \dots, q_{j-1}\}$ に直交する部分空間への直交射影となるので、

$$P_j = I - \sum_{i=1}^{j-1} q_i q_i^*$$

これより、

$$\begin{aligned} P_j a_j &= \left(I - \sum_{i=1}^{j-1} q_i q_i^*\right) a_j \qquad (m \times 1) \\ &= a_j - \sum_{i=1}^{j-1} q_i q_i^* a_j \\ &= a_j - \sum_{i=1}^{j-1} (q_i^* a_j) q_i \\ &= a_j - \sum_{i=1}^{j-1} r_{ij} q_i \quad (q_i^* a_i = r_{ij}) \end{aligned}$$

これは式 (7.5) と完全に一致する。これを正規化したものが、q_j の正体である。つまり、a_j に P_j を施すと a_j を $\{q_1, \dots, q_{j-1}\}$ に対して直交するベクトル v_j に変換する。

Modified Gram–Schmidt Algorithm

■ $P_j = P_{\perp q_{j-1}} \cdots P_{\perp q_2} P_{\perp q_1}$ **を示す。**

$$\begin{aligned} P_j &= P_{\perp q_{j-1}} \cdots P_{\perp q_2} P_{\perp q_1} \\ &= (I - q_{j-1} q_{j-1}^*) \cdots (I - q_2 q_2^*)(I - q_1 q_1^*) \\ &= \underline{I - \sum_{i=1}^{j-1} q_i q_i^*} \end{aligned}$$

■ **(8.4)-(8.6) を繋げる**

$$\begin{aligned} v_j &= P_j a_j \\ &= \underline{\left(I - \sum_{i=1}^{j-1} q_i q_i^* \right)} a_j \\ &= P_{\perp q_{j-1}} \cdots P_{\perp q_2} P_{\perp q_1} a_j \end{aligned}$$

つまり、式 (7.5) を以下のように考えられるということである。

$$\begin{aligned} v_j = v_j &= a_j - (q_1^* a_j) q_1 - \cdots - (q_{j-1}^* a_j) q_{j-1} \\ &= P_{\perp q_{j-1}} \cdots P_{\perp q_2} P_{\perp q_1} a_j \end{aligned}$$

Algorithm 7.1. Classical Gram–Schmidt (unstable)

for $j = 1$ to n
  $v_j = a_j$
  for $i = 1$ to $j - 1$
    $r_{ij} = q_i^* a_j$
    $v_j = v_j - r_{ij} q_i$
  $r_{jj} = \|v_j\|_2$
  $q_j = v_j / r_{jj}$

Algorithm 8.1. Modified Gram–Schmidt

for $i = 1$ to n
  $v_i = a_i$
for $i = 1$ to n
  $r_{ii} = \|v_i\|$
  $q_i = v_i / r_{ii}$
  for $j = i + 1$ to n
    $r_{ij} = q_i^* v_j$
    $v_j = v_j - r_{ij} q_i$

Modified Gram–Schmidt 法は以下の通り、

$$\begin{aligned} v_j &= a_j \\ &= P_{\perp q_1} a_j \\ &= P_{\perp q_2} P_{\perp q_1} a_j \\ &\vdots \\ &= P_{\perp q_{j-1}} \cdots P_{\perp q_2} P_{\perp q_1} a_j \end{aligned}$$

と掛け合わせていくアルゴリズムである。2 行目の計算結果に、$P_{\perp q_2}$ を掛けてと逐次的に行っていく。

$$
\begin{aligned}
v_j^{(1)} &= a_j \\
v_j^{(2)} &= P_{\perp q_1} v_j^{(1)} &&= v_j^{(1)} - q_1 q_1^* v_j^{(1)} \\
v_j^{(3)} &= P_{\perp q_2} v_j^{(2)} &&= v_j^{(2)} - q_2 q_2^* v_j^{(2)} \\
&\vdots &&\vdots \\
v_j = v_j^{(j)} &= P_{\perp q_{j-1}} v_j^{(j-1)} &&= v_j^{(j-1)} - q_{j-1} q_{j-1}^* v_j^{(j-1)}
\end{aligned}
$$

Classical では

$$v_n = a_n - \sum_{i=1}^{n-1} q_i^* a_j q_i$$

と一気に、a_n に対して $n-1$ 回分の減算を行ってから、$|v_n|$ を割ることで、q_n を求めている。

2.4 Householder Triangularization

2.4.1 Householder and Gram–Schmidt

式 (8.10) から Gram–Schmidt を用いて、

$$AR_1R_2\cdots R_n = \hat{Q}$$

と R を施していく形なら、reduced QR factorization of A となる。対して、Householder を用いて、

$$Q_n\cdots Q_2Q_1A = R$$

と計算していくと、full QR factorization of A となる。

2.4.2 The Better of Two Reflectors

x に対して、なるべく遠い方に鏡映したほうが安定するので、

$$v = -sign(x_1)\|x\|_2 e_1 - x \quad (= v') \tag{2.1}$$

となる。ただし、

$$v = sign(x_1)\|x\|_2 e_1 + x \quad (= -v') \tag{2.2}$$

でも問題ない。v は行列 F を構成する要素だが、

$$F = I - 2\frac{(-v)(-v)^*}{(-v)^*(-v)} = I - 2\frac{vv^*}{v^*v}$$

と符号同士が打ち消されるからである。そのため、アルゴリズムでは式 (2.2) を用いている。

アルゴリズムの

$$
\begin{aligned}
v_k &= v_k/\|v_k\|_2 \\
A_{k:m,k:n} &= A_{k:m,k:n} - 2v_k(v_k^* A_{k:m,k:n})
\end{aligned}
$$

について。

$$Q_k A = \begin{bmatrix} I & 0 \\ 0 & F \end{bmatrix} \begin{bmatrix} A_{1:k-1,1:n} \\ A_{k:m,k:n} \end{bmatrix} = \begin{bmatrix} A_{1:k-1,1:n} \\ FA_{k:m,k:n} \end{bmatrix}$$

行列 Q を構成する単位行列は $(k-1)\times(k-1)$ であるから、小行列 A はかけられて、$(k-1)\times n$ となる。また、$Q_k A$ の結果からわかるように、小行列 A の $(k-m+1)\times(k-m+1)$ が値が変わり、$FA_{k:m,k:n}$ となる。そのため、アルゴリズム的にはこの範囲の値のみを更新すれば良いこととなる。さらに、前の行で v の正規化を行っているため、

$$A_{k:m,k:n} = A_{k:m,k:n} - 2v_k(v_k^* A_{k:m,k:n})$$

となっている。**しかし、**$\|v_k\|_2$ を計算するのは平方根が出現するため、安定しない。教科書としてわかりやすくしている説がある。実際に組む場合は以下のような手順で計算されるのがふさわしい。

for $k = 1$**to** n
$x = A_{k:m,n}$
$v_k = sign(x_1)\|x\|_2 e_1 + x$
$\tau = 2/v^* v$
$w = \tau \cdot (v^* A_{k:m,k:n})$
$A_{k:m,k:n} = A_{k:m,k:n} - vw$

がアルゴリズムとして良い (Chat GPT 曰く)。これは計算量においても優れている。注目すべきは、

$$2\frac{vv^*}{v^*v}A$$

の計算手順である。

計算手順	掛け合わせ	計算量
$\frac{2}{v^*v}\cdot vv^*\cdot A$	スカラー値 × 行列 × 行列	$O(n^3)$
$v\cdot\frac{2}{v^*v}\cdot v^*A$	ベクトル × スカラー × 行ベクトル	$O(n^2)$

vv^* は行列となるため、行列同士の演算となり、計算量が膨大となる。加えて、これにスカラー値を掛けるとなると、積算回数は n^2 となる。対して、v^*A は行ベクトルであるから、$O(n^2)$ で抑えられる。また、スカラー値は列数の n 回分と優れる。

$$A = \begin{bmatrix} 3 & 0 & 1 \\ 4 & 5 & 2 \\ 0 & 4 & 3 \end{bmatrix} \quad (m = n = 3)$$

を用いて追っていく。

■ $k = 1$ のとき

$$x = A_{1:3,1} = \begin{bmatrix} 3 \\ 4 \\ 0 \end{bmatrix}$$

$$v = sign(x_1)\|x\|_2 e_1 + x = sign(3) \cdot 5 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 3 \\ 4 \\ 0 \end{bmatrix} = \begin{bmatrix} 8 \\ 4 \\ 0 \end{bmatrix}$$

$$\tau = 2/\|v\|_2^2 = 2/80 = 1/40$$

$$w = \tau \cdot (v \cdot A_{1:3,1:3}) = 1/40 \cdot [8 \quad 4 \quad 0] \begin{bmatrix} 3 & 0 & 1 \\ 4 & 5 & 2 \\ 0 & 4 & 3 \end{bmatrix} = [1 \quad 1/2 \quad 2/5]$$

$$A_{1:3,1:3} = A_{1:3,1:3} - vw = \begin{bmatrix} 3 & 0 & 1 \\ 4 & 5 & 2 \\ 0 & 4 & 3 \end{bmatrix} - \begin{bmatrix} 8 \\ 4 \\ 0 \end{bmatrix} [1 \quad 1/2 \quad 2/5] = \begin{bmatrix} -5 & -4 & -11/5 \\ 0 & 3 & 2/5 \\ 0 & 4 & 3 \end{bmatrix}$$

■ $k = 2$ のとき

$$x = A_{2:3,2} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

$$v = sign(x_1)\|x\|_2 e_1 + x = sign(3) \cdot 5 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 8 \\ 4 \end{bmatrix}$$

$$\tau = 2/\|v\|_2^2 = 2/80 = 1/40$$

$$w = \tau \cdot (v^{\mathsf{T}} \cdot A_{2:3,2:3}) = \frac{1}{40} \cdot [8 \quad 4] \begin{bmatrix} 3 & 2/5 \\ 4 & 3 \end{bmatrix}$$

$$= \frac{1}{40} \cdot [40 \quad 76/5] = [1 \quad 19/50]$$

$$A_{2:3,2:3} = A_{2:3,2:3} - vw = \begin{bmatrix} 3 & 2/5 \\ 4 & 3 \end{bmatrix} - \begin{bmatrix} 8 \\ 4 \end{bmatrix} [1 \quad 19/50] = \begin{bmatrix} -5 & -66/25 \\ 0 & 37/25 \end{bmatrix}$$

以上より、

$$A = R = \begin{bmatrix} -5 & -4 & -11/5 \\ 0 & -5 & -66/25 \\ 0 & 0 & 37/25 \end{bmatrix}$$

Listing 2.1: 自作 Householder

```python
import numpy as np
import numpy.linalg as LA

A = np.array([[3,0,1],[4,5,2],[0,4,3]], dtype=float)
# 2行2列以降の小行列を取り出す A[1:,1:]
# A.shape[0] 行数
# A.shape[1] 列数
for i in range(A.shape[0]-1):
    x = A[i:,[i]]
    e = np.identity(x.shape[0]) # 単位行列
    v = np.sign(x[0,0])*LA.norm(x)*e[:,[0]]+x
    #列ベクトルなので、x[0] = [3] , x[0,0] = 3
    t = 2/np.sum(v**2)
    w = t* v.T@A[i:,i:]
    A[i:,i:] -= v@w
print(A)

"""
A = [[3, 0, 1],
     [4, 5, 2],
     [0, 4, 3]]

A[:, 0]   → [3, 4, 0]       ← 1次元 shape=(3,)
A[:, [0]] → [[3],
             [4],
             [0]]           ← 2次元 shape=(3,1)

"""
```

Applying or Forming Q

Q は構成する v, τ などを求まる度に、使用すれば、行列サイズの掛け算が起こらない。

$$Qx \ : \ x \leftarrow (I - \tau_k v_k v_k^*)x \qquad (k = 1, \dots, n-1)$$

また、

$$Q_n \cdots Q_2 Q_1 A = R$$

であった。ならば、掛ける対象を単位行列 I にしてあげれば、

$$Q_n \cdots Q_2 Q_1 I = Q^*$$

となる。この手法は上三角行列が生成されるので、最後転置してあげれば良い。行列 A から行列 R を作ったときは、A[i:,i:] のように小行列に対して計算をしていた。同じ理論を行列 Q に対しても採用すればよいとわかる。

Listing 2.2: Q の計算に対応 (ChatGPT)

```
""" code omitted """
A = np.array([[3,0,1],[4,5,2],[0,4,3]], dtype=float)
A0 = A.copy()
Q = np.identity(A.shape[0])

for i in range(A.shape[0]-1):
""" code omitted """

""" code omitted """
    A[i:,i:] -= v @ w

    # Q を A の下に格納する
    wq = t * v.T @ Q[i:,:]
    Q[i:,:] -= v @ wq

R = A
Q = Q.T

print("Q=\n", Q)
print("R=\n", R)
print("Q@R=\n", Q @ R)
print("A0=\n", A0)
print("Q^T Q=\n", Q.T @ Q)
```

$$Q = \begin{bmatrix} -0.6 & 0.48 & 0.64 \\ -0.8 & -0.36 & -0.48 \\ 0. & -0.8 & 0.6 \end{bmatrix}$$

$$R = \begin{bmatrix} 3. & 0. & 1. \\ 4. & 5. & 2. \\ 0. & 0. & 1.48 \end{bmatrix}$$

$$QR = \begin{bmatrix} 3.00000000e+00 & -6.66133815e-16 & 1.00000000e+00 \\ 4.00000000e+00 & 5.00000000e+00 & 2.00000000e+00 \\ 0.00000000e+00 & 4.00000000e+00 & 3.00000000e+00 \end{bmatrix}$$

$$Q^*Q = \begin{bmatrix} 1.00000000e+00 & -1.39444012e-16 & -1.11910481e-16 \\ -1.39444012e-16 & 1.00000000e+00 & 1.95399252e-16 \\ -1.11910481e-16 & 1.95399252e-16 & 1.00000000e+00 \end{bmatrix}$$