Ayudantía
SCT - Semana - 10

Mínimos Cuadrados - Ecuaciones Normales y Factorización QR

1. Use mínimos cuadrados, las ecuaciones normales, el proceso de Gram-Schmidt y la factorización QR para encontrar la ecuación lineal $y = ax + b$ que mejor ajuste los tres puntos $(1, 3)$, $(2, 4)$ y $(3, 4)$.

Mínimos cuadrados

La recta $ax + b = y$ debería pasar por los puntos $(1, 3)$, $(2, 4)$ y $(3, 4)$. Sustituyendo las coordenadas x e y en la ecuación de la recta, se obtiene 3 ecuaciones lineales sobre los 2 coeficientes desconocidos a y b:

$$\begin{cases} a + b = 3 \\ 2a + b = 4 \\ 3a + b = 4 \end{cases}$$

Sin embargo, no es posible que la línea pase por los tres puntos al mismo tiempo, debido a que no están alineados. Por lo tanto, se debe realizar regresión lineal para estimar una línea que mejor ajuste estos tres puntos.

El enfoque a usar será el del álgebra lineal. Se puede expresar el sistema anterior como la siguiente ecuación vectorial:

$$\begin{pmatrix} a+b \\ 2a+b \\ 3a+b \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 4 \end{pmatrix} \iff a\underbrace{\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}}_{\mathbf{x}} + b\underbrace{\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}}_{\mathbf{1}} = \underbrace{\begin{pmatrix} 3 \\ 4 \\ 4 \end{pmatrix}}_{\mathbf{y}} \iff \underbrace{\begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} a \\ b \end{pmatrix}}_{\mathbf{k}} = \underbrace{\begin{pmatrix} 3 \\ 4 \\ 4 \end{pmatrix}}_{\mathbf{y}}$$

En los apuntes, se usa $A\mathbf{x} = \mathbf{b}$ en vez de $A\mathbf{k} = \mathbf{y}$, pero considero que, en este caso particular, $A\mathbf{k} = \mathbf{y}$ lleva a menos confusión.

En la ecuación anterior, se tiene una combinación lineal de los vectores $\mathbf{x} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$ y $\mathbf{1} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$. Las combinaciones lineales de dos vectores linealmente independientes forman un **plano**. Se busca la combinación $A\mathbf{k} = a\mathbf{x} + b\mathbf{1}$ en el plano que mejor aproxime el vector $\mathbf{y} = \begin{pmatrix} 3 \\ 4 \\ 4 \end{pmatrix}$. Para ello, se debe minimizar la distancia o **norma-2** dada por el vector $\mathbf{r} = \mathbf{y} - A\mathbf{k}$.

Ecuaciones normales

La distancia entre $\mathbf{y}$ y $A\mathbf{k}$ se minimiza cuando el vector $\mathbf{r}$ es **perpendicular al plano**. Para ello, $\mathbf{r}$ debe ser perpendicular a $\mathbf{x}$ y a $\mathbf{1}$, para así ser perpendicular a todas las combinaciones lineales $a\mathbf{x} + b\mathbf{1}$ en el plano. Es decir, su producto punto con $\mathbf{x}$ y con $\mathbf{1}$ debe ser 0:

$$\begin{cases} \mathbf{x}^T\mathbf{r} = 0 \\ \mathbf{1}^T\mathbf{r} = 0 \end{cases} \iff \begin{cases} \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}\mathbf{r} = 0 \\ \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}\mathbf{r} = 0 \end{cases} \iff \underbrace{\begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \end{pmatrix}}_{A^T}\mathbf{r} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Al expresar el sistema como una matriz, los vectores traspuestos $\mathbf{x}^T$ y $\mathbf{1}^T$ forman la matriz traspuesta A^T, por lo que el sistema queda como $A^T\mathbf{r} = \mathbf{0}$. Al reemplazar $\mathbf{r} = \mathbf{y} - A\mathbf{k}$, se obtiene:

$$\begin{aligned} A^T\mathbf{r} &= \mathbf{0} \\ A^T(\mathbf{y} - A\mathbf{k}) &= \mathbf{0} \\ A^T\mathbf{y} - A^TA\mathbf{k} &= \mathbf{0} \\ A^T\mathbf{y} &= A^TA\mathbf{k} \\ A^TA\mathbf{k} &= A^T\mathbf{y} \\ \begin{pmatrix} 14 & 6 \\ 6 & 3 \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix} &= \begin{pmatrix} 23 \\ 11 \end{pmatrix} \end{aligned}$$

Lo anterior se conoce como **"ecuaciones normales"**, debido a que se obtuvieron al hacer que el vector **r** fuera perpendicular o **normal** al plano.

Este es un sistema de 2x2 que se puede resolver de la forma tradicional, obteniendo las soluciones $a = \frac{1}{2}$ y $b = \frac{8}{3}$. Por lo tanto, la recta $y = ax+b$ que mejor estima los tres puntos originales es $y = \frac{1}{2}x+\frac{8}{3}$.

Sin embargo, para sistemas de ecuaciones más grandes, el número de condición del sistema suele ser muy alto. Para resolverlo, existe un método de resolución que toma ventaja de la estructura A^TA para reducir el número de condición: **la factorización QR. Para explicarlo, debemos entender el proceso de Gram-Schmidt.**

Proceso de Gram-Schmidt

El producto A^TA corresponde a hacer productos punto entre las columnas de A, las cuales corresponden a los vectores $\mathbf{x} = \begin{pmatrix}1\\2\\3\end{pmatrix}$ y $\mathbf{1} = \begin{pmatrix}1\\1\\1\end{pmatrix}$:

$$A^TA = \begin{pmatrix} — & \mathbf{x}^T & — \\ — & \mathbf{1}^T & — \end{pmatrix} \begin{pmatrix} | & | \\ \mathbf{x} & \mathbf{1} \\ | & | \end{pmatrix} = \begin{pmatrix} \mathbf{x}^T\mathbf{x} & \mathbf{x}^T\mathbf{1} \\ \mathbf{1}^T\mathbf{x} & \mathbf{1}^T\mathbf{1} \end{pmatrix}$$

pero el caso óptimo sería que las columnas $\mathbf{v}_1$ y $\mathbf{v}_2$ fueran ortogonales entre sí ($\mathbf{v}_1^T\mathbf{v}_2 = 0$) y tuvieran norma 1 ($\mathbf{v}_1^T\mathbf{v}_1 = \mathbf{v}_2^T\mathbf{v}_2 = 1$), pues así el producto $A^TA = \begin{pmatrix} \mathbf{v}_1^T\mathbf{v}_1 & \mathbf{v}_1^T\mathbf{v}_2 \\ \mathbf{v}_2^T\mathbf{v}_1 & \mathbf{v}_2^T\mathbf{v}_2 \end{pmatrix}$ daría la matriz identidad $I = \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}$.

Como, en general, las bases de **vectores ortonormales** (unitarios y ortogonales entre sí) son muy convenientes, vamos a usar el **proceso de Gram-Schmidt**, que toma una base de vectores y genera otra base ortonormal que genera el mismo espacio vectorial original.

Para los dos vectores $\mathbf{v}_1 = \mathbf{x}$ y $\mathbf{v}_2 = \mathbf{1}$, el proceso de Gram-Schmidt es sencillo y genera dos vectores ortonormales $\mathbf{q}_1$ y $\mathbf{q}_2$ de la siguiente manera:

a) Normalizar $\mathbf{v}_1$ para obtener $\mathbf{q}_1$.

La norma de $\mathbf{v}_1 = \mathbf{x} = \begin{pmatrix}1\\2\\3\end{pmatrix}$ es $\|\mathbf{v}_1\| = \sqrt{1^2+2^2+3^2} = \sqrt{14}$.

De esta manera, $\mathbf{q}_1 = \frac{1}{\sqrt{14}}\mathbf{v}_1 = \frac{1}{\sqrt{14}}\begin{pmatrix}1\\2\\3\end{pmatrix}$.

b) Hacer que $\mathbf{v}_2$ sea ortogonal a $\mathbf{q}_1$, y normalizar el resultado para obtener $\mathbf{q}_2$.

$\mathbf{v}_2$ se puede descomponer en la suma de dos componentes: uno ($\mathbf{v}_{2,\parallel}$)paralelo a $\mathbf{q}_1$, y otro ($\mathbf{v}_{2,\perp}$) perpendicular. Se debe eliminar su componente paralelo para obtener un vector puramente perpendicular. Este componente paralelo es:

$$\begin{aligned}\mathbf{v}_{2,\parallel} &= (\|\mathbf{v}_2\|\cos(\theta))\mathbf{q}_1 \\ &= (\|\mathbf{q}_1\|\|\mathbf{v}_2\|\cos(\theta))\mathbf{q}_1 \\ &= \left(\mathbf{q}_1^T\mathbf{v}_2\right)\mathbf{q}_1 \\ &= \frac{6}{\sqrt{14}}\mathbf{q}_1 \\ &= \frac{3}{7}\begin{pmatrix}1\\2\\3\end{pmatrix}\end{aligned}$$

Se resta este componente paralelo a $\mathbf{v}_2$ para obtener la parte perpendicular:

$$\begin{aligned}\mathbf{v}_{2,\perp} &= \mathbf{v}_2 - \mathbf{v}_{2,\parallel} \\ &= \begin{pmatrix}1\\1\\1\end{pmatrix} - \frac{3}{7}\begin{pmatrix}1\\2\\3\end{pmatrix} \\ &= \begin{pmatrix}4/7\\1/7\\-2/7\end{pmatrix} \\ &= \frac{1}{7}\begin{pmatrix}4\\1\\-2\end{pmatrix}\end{aligned}$$

y este componente perpendicular se normaliza para obtener $\mathbf{q}_2$. La norma de este componente es $\frac{1}{7}\sqrt{4^2+1^2+(-2)^2} = \frac{\sqrt{21}}{7}$. Dividiendo el vector anterior por $\frac{\sqrt{21}}{7}$, se obtiene el vector unitario $\mathbf{q}_2 = \frac{1}{\sqrt{21}}\begin{pmatrix}4\\1\\-2\end{pmatrix}$.

Factorización QR

a) Al normalizar $\mathbf{v}_1$ para obtener $\mathbf{q}_1$, se puede observar que:

$$\mathbf{v}_1 = \sqrt{14}\mathbf{q}_1$$

que se puede expresar así:

$$\begin{aligned}\mathbf{v}_1 &= \sqrt{14}\mathbf{q}_1 + 0\mathbf{q}_2 \\ &= \begin{pmatrix}| & |\\ \mathbf{q}_1 & \mathbf{q}_2\\ | & |\end{pmatrix}\begin{pmatrix}\sqrt{14}\\0\end{pmatrix}\end{aligned}$$

b) En el proceso de obtener $\mathbf{q}_2$, se puede ver que el componente de $\mathbf{v}_2$ paralelo a $\mathbf{q}_1$ es $\mathbf{v}_{2,\parallel} = \frac{6}{\sqrt{14}}\mathbf{q}_1$ y su componente perpendicular se normaliza para obtener $\mathbf{q}_2$, de manera que $\mathbf{v}_{2,\perp} = \frac{\sqrt{21}}{7}\mathbf{q}_2$. Sumando ambos:

$$\begin{aligned}\mathbf{v}_2 &= \frac{6}{\sqrt{14}}\mathbf{q}_1 + \frac{\sqrt{21}}{7}\mathbf{q}_2 \\ &= \begin{pmatrix}| & |\\ \mathbf{q}_1 & \mathbf{q}_2\\ | & |\end{pmatrix}\begin{pmatrix}\frac{6}{\sqrt{14}}\\ \frac{\sqrt{21}}{7}\end{pmatrix}\end{aligned}$$

Combinando los vectores $\mathbf{v}_1$ y $\mathbf{v}_2$ en la matriz A, se llega a su **factorización QR**:

$$\underbrace{\begin{pmatrix}| & |\\ \mathbf{v}_1 & \mathbf{v}_2\\ | & |\end{pmatrix}}_{A} = \underbrace{\begin{pmatrix}| & |\\ \mathbf{q}_1 & \mathbf{q}_2\\ | & |\end{pmatrix}}_{Q}\underbrace{\begin{pmatrix}\sqrt{14} & \frac{6}{\sqrt{14}}\\ 0 & \frac{\sqrt{21}}{7}\end{pmatrix}}_{R}$$

Las columnas $\mathbf{q}_1$ y $\mathbf{q}_2$ de la matriz Q son ortonormales, lo que causa que

$$Q^TQ = \begin{pmatrix}— & \mathbf{q}_1^T & —\\ — & \mathbf{q}_2^T & —\end{pmatrix}\begin{pmatrix}| & |\\ \mathbf{q}_1 & \mathbf{q}_2\\ | & |\end{pmatrix} = \begin{pmatrix}\mathbf{q}_1^T\mathbf{q}_1 & \mathbf{q}_1^T\mathbf{q}_2\\ \mathbf{q}_2^T\mathbf{q}_1 & \mathbf{q}_2^T\mathbf{q}_2\end{pmatrix} = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} = I_2$$

Además, R es una matriz cuadrada triangular superior.

Esto tiene consecuencias interesantes cuando regresamos a las ecuaciones normales:

$$A^TA\mathbf{k} = A^T\mathbf{y}$$

Si se sustituye $A = QR$, el producto $A^T A$ se convierte en $R^T Q^T QR$ y se reduce a simplemente $R^T R$, obteniendo:

$$\begin{aligned} A^T A\mathbf{k} &= A^T\mathbf{y} \\ R^T Q^T QR\mathbf{k} &= R^T Q^T\mathbf{y} \\ R^T R\mathbf{k} &= R^T Q^T\mathbf{y} \end{aligned}$$

R y R^T son matrices no singulares, puesto que sus diagonales contienen las normas de los vectores ortogonalizados antes de ser normalizados. Si los vectores originales $\mathbf{v}_i$ son linealmente independientes, esto siempre ocurre. Se puede cancelar R^T a ambos lados multiplicando por su inversa:

$$\begin{aligned} R^T R\mathbf{k} &= R^T Q^T\mathbf{y} \\ R\mathbf{k} &= Q^T\mathbf{y} \end{aligned}$$

Esto ahorra dos multiplicaciones de matrices y simplifica el sistema.

Para finalizar, como R es una matriz triangular superior, el sistema se puede solucionar mediante *backward substitution* (sustitución hacia atrás).

Solución

$$\begin{aligned} R\mathbf{k} &= Q^T\mathbf{y} \\ \begin{pmatrix} \sqrt{14} & \frac{6}{\sqrt{14}} \\ 0 & \frac{\sqrt{21}}{7} \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} &= \begin{pmatrix} \frac{1}{\sqrt{14}} & \frac{2}{\sqrt{14}} & \frac{3}{\sqrt{14}} \\ \frac{4}{\sqrt{21}} & \frac{1}{\sqrt{21}} & \frac{-2}{\sqrt{21}} \end{pmatrix} \begin{pmatrix} 3 \\ 4 \\ 4 \end{pmatrix} \\ \begin{pmatrix} \sqrt{14} & \frac{6}{\sqrt{14}} \\ 0 & \frac{\sqrt{21}}{7} \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} &= \begin{pmatrix} \frac{23}{\sqrt{14}} \\ \frac{8}{\sqrt{21}} \end{pmatrix} \end{aligned}$$

Backward substitution:

a) De la última fila o ecuación, se obtiene el valor de b:

$$\begin{aligned} \frac{\sqrt{21}}{7}b &= \frac{8}{\sqrt{21}} \\ b &= \frac{7 \cdot 8}{21} \\ b &= \frac{8}{3} \end{aligned}$$

b) El valor de b se sustituye hacia atrás (backward substitution) en la ecuación anterior:

$$\begin{aligned} \sqrt{14}a + \frac{6}{\sqrt{14}}b &= \frac{23}{\sqrt{14}} \\ \sqrt{14}a + \frac{6}{\sqrt{14}} \cdot \frac{8}{3} &= \frac{23}{\sqrt{14}} \\ \sqrt{14}a + \frac{16}{\sqrt{14}} &= \frac{23}{\sqrt{14}} \\ \sqrt{14}a &= \frac{7}{\sqrt{14}} \\ a &= \frac{7}{14} \\ a &= \frac{1}{2} \end{aligned}$$

Por lo tanto, la recta que mejor ajusta los puntos (1, 3), (2, 4) y (3, 4) es $y = \frac{1}{2}x + \frac{8}{3}$**.**

2. Considere la siguiente secuencia de matrices tridiagonales $T_n \in \mathbb{R}^{(n+1)\times n}$,

$$T_n = \begin{bmatrix} a_1 & c_1 & 0 & \dots & \dots & \dots & 0 \\ b_1 & a_2 & c_2 & 0 & \dots & \dots & \vdots \\ 0 & b_2 & a_3 & c_3 & 0 & \dots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 & b_{n-2} & a_{n-1} & c_{n-1} \\ \vdots & \ddots & \ddots & \ddots & 0 & b_{n-1} & a_n \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 & b_n \end{bmatrix},$$

donde los coeficientes a_i, b_i y c_i son conocidos. Note que T_n no es una matriz cuadrada, dado que tiene una fila más que la cantidad de columnas. Considere que usted tiene acceso a la factorización QR reducida de la matriz T_n, es decir, tiene acceso a,

$$T_n = \widehat{Q}_n \, \widehat{R}_n = \begin{bmatrix} \mathbf{q}_1^{[n]} & \mathbf{q}_2^{[n]} & \dots & \mathbf{q}_n^{[n]} \end{bmatrix} \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & \dots & r_{1,n} \\ 0 & r_{2,2} & r_{2,3} & \dots & r_{2,n} \\ 0 & 0 & r_{3,3} & \dots & r_{3,n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & 0 & r_{n,n} \end{bmatrix},$$

donde $\widehat{Q}_n \in \mathbb{R}^{(n+1)\times n}$ y $\widehat{R}_n \in \mathbb{R}^{n\times n}$. En este caso, se usa el súper-índice "$[n]$" para denotar que el vector $\mathbf{q}_i^{[n]} \in \mathbb{R}^{n+1}$, y está asociado a la matriz T_n.

Como se indicó al principio, T_n denota una secuencia de matrices tridiagonales, esto significa que la siguiente matriz de esta secuencia es,

$$T_{n+1} = \begin{bmatrix} a_1 & c_1 & 0 & \dots & \dots & \dots & \dots & 0 \\ b_1 & a_2 & c_2 & 0 & \dots & \dots & \dots & \vdots \\ 0 & b_2 & a_3 & c_3 & 0 & \dots & \dots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots \ddots & 0 & \\ \vdots & \ddots & \ddots & 0 & b_{n-2} & a_{n-1} & c_{n-1} & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 & b_{n-1} & a_n & c_n \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 & b_n & a_{n+1} \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & b_{n+1} \end{bmatrix},$$

la cual se diferencia de T_n solamente porque se agregó una columna y una fila adicional. Este patrón puede representarse de la siguiente forma,

$$T_{n+1} = \left(\begin{array}{ccc|c} & & & 0 \\ & & & \vdots \\ & T_n & & 0 \\ & & & c_n \\ & & & a_{n+1} \\ \hline 0 & \dots & 0 & b_{n+1} \end{array} \right),$$

es decir, la mayor parte de la matriz T_{n+1} corresponde a la matriz T_n. La tarea a resolver es obtener la factorización QR reducida de la matriz T_{n+1} dado que conocemos la factorización QR reducida de la matriz T_n. Para su desarrollo considere

la siguiente identidad,

$$
\begin{aligned}
T_{n+1} &= \left(\begin{array}{ccc|c} & & & 0 \\ & T_n & & \vdots \\ & & & 0 \\ & & & c_n \\ & & & a_{n+1} \\ \hline 0 & \dots & 0 & b_{n+1} \end{array}\right) \\
&= \left(\begin{array}{ccc|c} & & & 0 \\ & \widehat{Q}_n\,\widehat{R}_n & & \vdots \\ & & & 0 \\ & & & c_n \\ & & & a_{n+1} \\ \hline 0 & \dots & 0 & b_{n+1} \end{array}\right) \\
&= \widehat{Q}_{n+1}\,\widehat{R}_{n+1} \\
&= \begin{bmatrix} \mathbf{q}_1^{[n+1]} & \mathbf{q}_2^{[n+1]} & \dots & \mathbf{q}_n^{[n+1]} & \mathbf{q}_{n+1}^{[n+1]} \end{bmatrix} \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & \dots & r_{1,n} & r_{1,n+1} \\ 0 & r_{2,2} & r_{2,3} & \dots & r_{2,n} & r_{2,n+1} \\ 0 & 0 & r_{3,3} & \dots & r_{3,n} & r_{2,n+1} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & 0 & r_{n,n} & r_{n,n+1} \\ 0 & \dots & \dots & 0 & 0 & r_{n+1,n+1} \end{bmatrix}.
\end{aligned}
$$

A partir de las identidades anteriores, usted debe determinar los vectores $\mathbf{q}_i^{[n+1]}$ para $i \in \{1, 2, 3, \dots, n+1\}$ y la última columna de R_{n+1}.

Recuerde que $\mathbf{q}_i^{[n]} \in \mathbb{R}^{n+1}$, pero $\mathbf{q}_i^{[n+1]} \in \mathbb{R}^{n+2}$. Es decir, tiene una diferencia de una unidad en la dimensión del espacio vectorial asociado. Lo importante es que esta diferencia puede ser resuelta convenientemente para obtener las primeras n columnas de la matriz $\widehat{Q}_{n+1}$.

- *Hint 1: You may consider useful this relationship:* $\mathbf{q}_j^{[n+1]} = [(\mathbf{q}_j^{[n]})^T, 0]^T$, *for* $j \in \{1, 2, 3, \dots, n\}$ *and* ^{T} *is the transpose operator.*
- *Hint 2: Notice that the coefficients* $r_{i,j}$ *do not have a super index, this means that the only missing part of* R_{n+1} *is actually its last column!*

Preguntas

(a) Explique teóricamente como obtendrá la factorización QR reducida de T_{n+1} a partir de la factorización QR reducida de T_n.

Según la pista 1, para todo $j \in \{1, 2, 3, ..., n\}$:

$$\mathbf{q}_j^{[n+1]} = \begin{pmatrix} \mathbf{q}_j^{[n]} \\ 0 \end{pmatrix}$$

Para poder calcular el último vector ortonormal $\mathbf{q}_{n+1}^{[n+1]}$ y la última columna de R, hay que terminar el proceso de Gram-Schmidt: se debe tomar la última columna de T_{n+1}, la cual llamaremos $\mathbf{t}_{\text{last}}$, hacerla ortogonal a todos los vectores anteriores $\mathbf{q}_j^{[n+1]}$ y normalizar el resultado, obteniendo así $\mathbf{q}_{n+1}^{[n+1]}$. Como resultado, $\mathbf{t}_{\text{last}}$ será una combinación lineal de los $n+1$ vectores ortonormales $\mathbf{q}_j^{[n+1]}$:

$$\mathbf{t}_{\text{last}} = \tilde{r}_1 \mathbf{q}_1^{[n+1]} + \tilde{r}_2 \mathbf{q}_2^{[n+1]} + ... + \tilde{r}_{n+1} \mathbf{q}_{n+1}^{[n+1]}$$

donde definimos $\tilde{r}_j$ como un alias más corto de $r_{j,n+1}$, es decir, un elemento de la última columna de R_{n+1}. Como tal, se cumple que

$$\tilde{r}_j = \mathbf{q}_j^{[n+1]} \cdot \mathbf{t}_{\text{last}}, \quad \forall j \in \{1, 2, ..., n\}$$

y, para $j = n+1$, $\tilde{r}_{n+1}$ es la norma del vector $\mathbf{t}_{\text{last}}$ después de restarle las componentes paralelas a los demás vectores de tal forma de volverlo ortogonal a ellos.

El método anterior es teóricamente válido y es directamente implementable en Python con NumPy: es simplemente terminar Gram-Schmidt. Basta con crear un vector $\mathbf{w} = \mathbf{t}_{\text{last}}$ y, por cada $j \in \{1, 2, ..., n\}$, calcular el

producto punto $\tilde{r}_j = \mathbf{q}_j^{[n+1]} \cdot \mathbf{w}$ y restar $\tilde{r}_j \mathbf{q}_j^{[n+1]}$ a $\mathbf{w}$ para ortogonalizarlo respecto de $\mathbf{q}_j$. Al finalizar el loop anterior, se calcula la norma $\tilde{r}_{n+1} = \|\mathbf{w}\|$ para normalizar $\mathbf{w}$ y obtener $\mathbf{q}_{n+1}^{[n+1]}$ junto con todos los $\tilde{r}_j$.
Sin embargo, lo anterior se puede optimizar: **no es necesario ortogonalizar a $\mathbf{t}_{\text{last}}$ respecto de todos los $\mathbf{q}_j^{[n+1]}$ para $j \leq n$, porque ya es ortogonal a todos los $\mathbf{q}_j^{[n+1]}$ para $j \leq n-2$.** El motivo de esto es la estructura tridiagonal de la matriz T_{n+1}. Gracias a ella, si uno realiza a mano el proceso de Gram-Schmidt, notará que los vectores ortonormales $\mathbf{q}_j^{[n+1]}$ obtenidos tienen esta forma:

$$\mathbf{q}_1^{[n+1]} = \begin{pmatrix} \times \\ \times \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{q}_2^{[n+1]} = \begin{pmatrix} \times \\ \times \\ \times \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{q}_3^{[n+1]} = \begin{pmatrix} \times \\ \times \\ \times \\ \times \\ \vdots \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad \cdots \quad \mathbf{q}_{n-2}^{[n+1]} = \begin{pmatrix} \times \\ \times \\ \times \\ \times \\ \vdots \\ \times \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{q}_{n-1}^{[n+1]} = \begin{pmatrix} \times \\ \times \\ \times \\ \times \\ \vdots \\ \times \\ \times \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{q}_n^{[n+1]} = \begin{pmatrix} \times \\ \times \\ \times \\ \times \\ \vdots \\ \times \\ \times \\ \times \\ 0 \end{pmatrix},$$

donde las cruces ($\times$) representan valores que podrían no ser nulos. Si se sabe que

$$\mathbf{t}_{\text{last}} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ c_n \\ a_{n+1} \\ b_{n+1} \end{pmatrix},$$

entonces se puede apreciar que, para todo $j \in \{1, 2, ..., n-2\}$, el producto punto $\mathbf{q}_j^{[n+1]} \cdot \mathbf{t}_{\text{last}} = 0$, es decir, $\tilde{r}_1 = \tilde{r}_2 = ... = \tilde{r}_{n-2} = 0$, sin necesidad de calcular nada.
Esto simplifica el algoritmo propuesto anteriormente: basta con ortogonalizar $\mathbf{t}_{\text{last}}$ respecto de $\mathbf{q}_{n-1}^{[n-1]}$ y $\mathbf{q}_n^{[n-1]}$, calculando $\tilde{r}_{n-1}$ y $\tilde{r}_n$. Luego, el vector resultante $\mathbf{w} = \mathbf{t}_{\text{last}} - \tilde{r}_{n-1}\mathbf{q}_{n-1}^{[n+1]} - \tilde{r}_n \mathbf{q}_n^{[n+1]}$ se divide entre su norma $\tilde{r}_n = \|\mathbf{w}\|$ para obtener $\mathbf{q}_{n+1}^{[n+1]}$. La última columna de R_{n+1} es:

$$\mathbf{r}_{\text{last}} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \tilde{r}_{n-1} \\ \tilde{r}_n \\ \tilde{r}_{n+1} \end{pmatrix}.$$

(b) Construya la función "'incrementalQR"' que obtiene la factorización QR reducida de la matriz T_{n+1} a partir de la factorización reducida de T_n. El input de esta función corresponde a Q_n, R_n, a_{n+1}, b_{n+1}, y c_n. IMPORTANTE: Esta función se considerará correcta si obtiene adecuadamente la factorización QR reducida de T_{n+1} a partir de la factorización QR reducida de T_n. Si se obtiene la factorización QR reducida directamente de la matriz T_{n+1}, no se considerará correcta. La importancia de esto es que obtener la factorización QR reducida de esta forma reduce significativamente la cantidad de operaciones elementales requerida, acelerando la computación significativamente.

```
'''
INPUT:
Qn    : (ndarray) Matriz unitaria Q_n de la factorización QR reducida de la matriz
        T_n = Q_n * R_n.
Rn    : (ndarray) Matriz triangular superior R_n de la factorización QR reducida de
```

```
        la matriz T_n = Q_n * R_n.
a     : (float)   Coeficiente a_{n+1} de la matriz T_{n+1}.
b     : (float)   Coeficiente b_{n+1} de la matriz T_{n+1}.
c     : (float)   Coeficiente c_n de la matriz T_{n+1}.

OUTPUT:
Q_next: (ndarray) Matriz unitaria Q_{n+1} de la factorización QR reducida de la
         matriz T_{n+1} = Q_{n+1} * R_{n+1}.
R_next: (ndarray) Matriz triangular superior R_{n+1} de la factorización QR
        reducida de la matriz T_{n+1} = Q_{n+1} * R_{n+1}.
'''
def incrementalQR(Qn, Rn, a, b, c):
    # Your own code.
    Q_next = np.zeros(2, 1) # Remover esta línea si usted implementa esta función.
    R_next = np.zeros(1, 1) # Remover esta línea si usted implementa esta función.
    return Q_next, R_next
```

```python
def incrementalQR(Qn, Rn, a, b, c):
    n = len(Qn)

    Q_next = np.zeros((n+1, n+1))
    Q_next[:n, :n] = Qn
    Q_next[n-1, n] = c
    Q_next[n, n] = a
    Q_next[n+1, n] = b

    R_next = np.zeros((n+1, n+1))
    R_next[:n, :n] = Rn

    # Ortogonalizar respecto de q_{n-1}^{[n+1]}, si es que existe.
    if n >= 2:
        R_next[n-2, n] = Q_next[n-2, n-2] * c
        Q_next[:, n] -= R_next[n-2, n] * Q_next[:, n-2]

    # Ortogonalizar respecto de q_{n}^{[n+1]}, si es que existe.
    if n >= 1:
        R_next[n-1, n] = Q_next[n-2, n-1] * c + Q_next[n-1, n-1] * a
        Q_next[:, n] -= R_next[n-1, n] * Q_next[:, n-1]

    # Normalizar.
    R_next[n, n] = np.linalg.norm(Q_next[:, n])
    Q_next[:, n] /= R_next[n, n]

    return Q_next, R_next
```