Exercici 4: Mètodes d'ortogonalització - QR

Álvaro Ortiz Villa

Àlgebra Lineal Numèrica (G. Matemàtiques, FME). Maig 2021

1.1) Al multiplicar la matriu H_i per una matriu A tal que $A_{i:m,1:i-1} = 0$ només modifica les últimes $m-i+1$ files i columnes de la matriu A. En particular, vegeu que $(H_iA)_{i:m,i:n} = P_iA_{i:m,i:n}$.

Sigui A una matriu $m \times n$, si multipliquem per blocs obtenim

$$H_iA = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_i \end{bmatrix} \times \begin{bmatrix} \mathbf{A}_{1:i-1,1:i-1} & \mathbf{A}_{1:i-1,i,n} \\ \mathbf{0} & \mathbf{A}_{i:m,i:n} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{1:i-1,1:i-1} & \mathbf{A}_{1:i-1,i,n} \\ \mathbf{0} & \mathbf{P}_i\mathbf{A}_{i:m,i:n} \end{bmatrix}$$

Aleshores, $(H_iA)_{i:m,i:n} = P_iA_{i:m,i:n}$.

1.2) Si es considera $P_i = I - 2\dfrac{vv^T}{v^Tv}$ on $v = x + sgn(x_1)\|A\|e^1$ per $x = A_{i:m,i}$ i on e^1 és el vector de mida $m-i+1$ tal que $e^1 = [1, 0, \cdots, 0]^T$, comproveu que la submatriu $(H_iA)_{i:m,i:n} = P_iA_{i:m,i:n}$ té zeros sota la primera columna.

A classe hem vist que si prenem $v = x + sgn(x_1)\|x\|e^1$ aleshores $Px = \pm\|x\|\ e^1$, és a dir l'operació comprimeix tot el vector x a la primera component. Aleshores, si prenem com a vector $x = A_{i:m,i}$ la columna i-èssima de la matriu A des de la fila i-èssima fins a la darrera llavors $v_{i:m} = A_{i:m,i} \pm \|A_{i:m,i}\|e^1$, d'on

$$(H_iA)_{i:m,i} = P_iA_{i:m,i} = P_i \begin{bmatrix} \mathrm{a}_{i,i} \\ \mathrm{a}_{i+1,i} \\ \vdots \\ \mathrm{a}_{m,i} \end{bmatrix} = \begin{bmatrix} \pm\|A_{i:m,i}\| \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

i per tant la submatriu $(H_iA)_{i:m,i:n}$ té zeros sota la primera columna.

2.1) Definiu una funció `matriuHouseholder(A, i)` que donada una matriu A i un índex i, retorni la matriu de Householder H_i.

Adaptem el càlcul de la matriu a python per a $j = 0, 1, 2, ...(j := i-1)$:

Codi 1 – Factorització QR Householder: matriu H en el pas j-èssim

```
1  def matriuHouseholder(A, i):
2      m = len(A)
3      I = np.eye(i)
4      x = A[i:m, i]
5      v = np.copy(x).reshape(len(x), 1)
6      v[0] = v[0]+np.sign(x[0])*np.linalg.norm(x)
7      P = np.eye(m-i)-2*(v@np.transpose(v))/(np.transpose(v)@v)
8      H = np.block([[I, np.zeros((i, m-i))],
9                    [np.zeros((m-i, i)), P]])
10 return H
```

2.2) Genereu l'`script` `main1_steps_householder.py` que mostri pas per pas com es calcula la factorització de QR reduïda utilitzant les matrius de Householder de la matriu que s'indica.

Executant el codi a l'`script` obtenim esquemàticament les següents transformacions de la matriu A:

$$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 7\\ 4 & 2 & 3\\ 4 & 2 & 2\end{bmatrix} \quad \mathbf{A}$$

$$\xrightarrow{H_1} \quad \begin{bmatrix}-9.89 & -9.49 & -9.69\\ 0 & 0.78 & 1.34\\ 0 & 0.62 & -1.15\\ 0 & -2.22 & -1.66\\ 0 & -2.22 & -2.66\end{bmatrix} \quad \mathbf{H_1A}$$

$$\xrightarrow{H_2} \quad \begin{bmatrix}-9.89 & -9.49 & -9.69\\ 0 & -3.29 & -3.01\\ 0 & 0 & -1.81\\ 0 & 0 & 0.71\\ 0 & 0 & -0.29\end{bmatrix} \quad \mathbf{H_2H_1A}$$

$$\xrightarrow{H_3} \quad \begin{bmatrix}-9.89 & -9.49 & -9.69\\ 0 & -3.29 & -3.01\\ 0 & 0 & 1.97\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix} \quad \mathbf{H_3H_2H_1A}$$

2.3) Definiu una funció `QRHouseholder(A,b)` que retorni una matriu R (matriu reduïda a triangular superior) obtinguda a partir de la matriu A després d'aplicar-li les matrius de Householder i el vector Q^Tb utilitzant l'algorisme que s'indica.

Codi 2 – Factorització QR Householder per a resolució de sistemes lineals

```
1  def QRHouseholder(A, b):
2      m,n = A.shape
3      for k in range(n):
4          x = A[k:m, k]
5          vk = np.copy(x).reshape(len(x), 1)
6          vk[0] = vk[0]+np.sign(x[0])*np.linalg.norm(x)
7          vk = vk/np.linalg.norm(vk)
8          A[k:m, k:n] = A[k:m, k:n]
9                        -2*vk@(np.transpose(vk)@(A[k:m, k:n]))
10         b[k:m] = b[k:m] -2*vk@(np.transpose(vk)@(b[k:m]))
11 return A, b
```

2.4) Genereu el fitxer `main2_linearSystem_QRHouseholder.py` que utilitzi `QRHouseholder` per trobar una solució x de $(Q^tA)_{1:n,1:n}x = R_{1:n,1:n}x = (Q^tb)_{1:n}$ amb A de 2.2. i $b = (1,\cdots,1)^T$. Comproveu l'exactitud del resultat com a solució del problema original. Quina és la causa del resultat observat?

Apliquem la funció `QRHouseholder(A,b)` i solucionem el sistema per a $R_{1:n,1:n}x = (Q^tb)_{1:n}$ d'on s'obté que $x^* = [0.179, -0.388, 0.406]^T$. Tanmateix, observem que $Ax^* \neq b$, de manera que el resultat no és exacte. Això es deu al fet que el sistema que considerem és sobredeterminat, és a dir conté més equacions que incògnites i per tant en general no existeix una única solució. L'objectiu en la resolució ha estat trobar un x^* tal que minimitzi el quadrat del residu: $x^* = \arg\min_{\vec{x}\in\mathbb{R}^n} \|Ax - b\|^2$. A classe hem vist que, seguint aquest algorisme, la solució obtinguda proporciona el mínim residu possible. En aquest cas, calculem que el residu és $\|Ax^* - b\| = 0.523$.

3.1) Plantejeu el sistema de les equacions normals i calculeu els coeficients del polinomi. Calculeu-lo en un script `main3_temperatures.py` Quin és el número de

condició de la matriu?

El sistema d'equacions normals és $A^TAc = A^Tb$ on $c^* = \arg\min_{\vec{x}\in\mathbb{R}^n} \|Ac-b\|^2$. Observem que la matriu A és una matriu de Vandermonde, i per això s'ha programat de dues maneres: primer amb un bucle per files i per columnes, i segon amb la comanda `np.flip(np.vander(year, 6), 1)`. La solució de mínims quadrats del sistema sobredeterminat difereix significativament segons el mètode ja que obtenim respectivament:

$$c^* = [2.36 \times 10^5, -4.70 \times 10^2, 3.45 \times 10^{-1}, -1.06 \times 10^{-4}, 8.79 \times 10^{-9}, 9.84 \times 10^{-13}]^T$$

$$c^* = [1.39 \times 10^5, -2.28 \times 10^2, -9.97 \times 10^{-2}, 1.78 \times 10^{-5}, -2.23 \times 10^{-8}, 4.12 \times 10^{-12}]^T$$

Ara bé, quan avaluem els dos polinomis notem que la diferència és prou petita i a més obtenim un residu similar de $\|Ac^* - b\| = 1.32$. En qualsevol cas, veiem el nombre de condició de la matriu és molt elevat –al voltant de 3.69×10^{37}–, cosa que ens diu que la matriu A és mal condicionada.

3.2) Alternativament, resolgueu el sistema lineal de l'Eq. (7) amb el codi desenvolupat a la secció anterior.

Apliquem els apartats 2.3 i 2.4 en aquest sistema i obtenim la següent estimació dels coeficients del polinomi:

$$c^* = [9.49 \times 10^6, -2.42 \times 10^4, 2.47 \times 10^1, -1.26 \times 10^{-2}, 3.19 \times 10^{-6}, -3.26 \times 10^{-10}]^T$$

En aquest cas, l'estimació proporciona un residu lleugerament inferior al de l'estimació per equacions normals: $\|Ac^* - b\| = 1.29$

3.3) Dibuixeu les dades i els polinomis obtinguts als apartats anteriors. S'obté el mateix polinomi amb les dues estratègies? Interpreteu els resultats.

La gràfica 1 il·lustra el polinomi estimat pels dos mètodes. Clarament notem que no s'obté el mateix polinomi, tot i que ambdós s'ajusten bastant a les dades i són força similars. En particular, la norma de la diferència entre els dos vectors de coeficients és de 0.276. A simple vista sembla que la diferència entre les dues estimacions és més gran a major dispersió de les dades.

3.4) En un nou fitxer script `main4_temperaturesEscalades.py` Feu un escalat de les dades amb el canvi de variable que es proposa i repetiu els apartats anteriors. Aporta algun benefici escalar les dades?

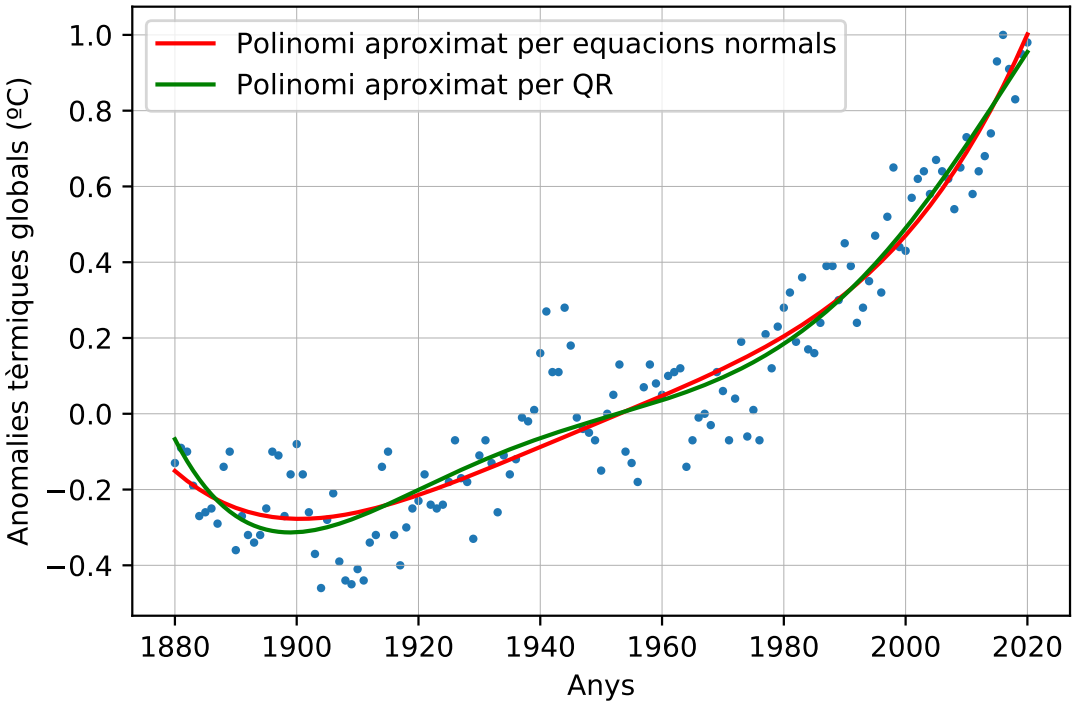
Amb el canvi d'escala a la variable temps obtenim una estimació gairebé idèntica entre l'estratègia d'equacions normals i la factorització QR:

$$c^* = [-0.07, -4.39, 25.58, -54.87, 52.24, -17.53]^T$$

A la gràfica 2 veiem l'aproximació del polinomi seguint els dos mètodes. Ara el número de condició la matriu és molt inferior (13998385) i si considerem la norma de la diferència entre els dos vectors de coeficients és significativament menor, d'aproximadament 2×10^{-11}. Pel que fa als residus de les estimacions, són força similars a l'estimació QR amb les dades sense escalar.

3.5) Estimeu l'anomalia tèrmica a l'any 2030. Quin de les dues metodologies (dades escalades o no) has triat per usar i perquè? Hem vist que l'aproximació del polinomi per equacions normals i per la factorització QR produeix estimacions molt semblants quan s'escalen les dades. A més, el residu $\|Ac^* - b\|$ és en conjunt més petit. Per tant, si avaluem el polinomi estimat amb les dades escalades a $p(2030)$ obtenim que la predicció d'anomalia tèrmica al 2030 és de 1.1895, cosa que fa alertar de l'evolució de la temperatura en el planeta en un futur pròxim.

Gràfica 1 – *Aproximació de polinomi per mínims quadrats (Dades originals)*



Gràfica 2 – *Aproximació de polinomi per mínims quadrats (Dades escalades)*