

Factorización  $A = QR$  (Int. to linear Algebra. Strang 239)

$$A = \begin{bmatrix} 1 & 1 & 1 \\ a & b & c \\ 1 & 1 & 1 \end{bmatrix} \rightarrow Q = \begin{bmatrix} 1 & 1 & 1 \\ q_1 & q_2 & q_3 \\ 1 & 1 & 1 \end{bmatrix}$$

En la clase anterior vimos cómo pensar de  $A \rightarrow Q$ . Busquemos ahora la forma matricial  $A = QR$ .

Al desarrollar Gram-Schmidt vimos que  $q_i$  eran comb. lineales de  $a, b$  y  $c$  (por construcción). De hecho:

- \*  $a, A$  y  $q_1$  están sobre una linea
- \*  $a, b$  y  $A, B$  y  $q_1, q_2$  están sobre un plano
- \*  $a, b, c$  y  $A, B, C$  y  $q_1, q_2, q_3$  están en un subespacio (dim. 3).

En cada paso  $a_1, \dots, a_K$  son combinaciones de  $q_1, \dots, q_K$ , los  $q$  posteriores ( $q_{K+1}, q_{K+2}, \dots$ ) no están implicados.

En forma matricial, esto significa que  $R$  es de la forma:

$$\begin{bmatrix} 1 & 1 & 1 \\ a & b & c \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ q_1 & q_2 & q_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} q_1^T a & q_1^T b & q_1^T c \\ 0 & q_2^T b & q_2^T c \\ 0 & 0 & q_3^T c \end{bmatrix}$$

$$A = Q R$$

En realidad, todo es más sencillo si le damos la forma  
al problema. Como  $Q$  es matriz ortogonal  $\rightarrow Q^{-1} = Q^T$  y

$$Q^T A = R$$

$$\begin{matrix} Q^T \\ \left[ \begin{array}{c|c|c} q_1 & | & | \\ q_2 & a & b \\ q_3 & | & c \end{array} \right] \\ \hline \end{matrix} = \begin{matrix} A \\ \left[ \begin{array}{c|c|c} | & | & | \\ a & b & c \\ | & | & | \end{array} \right] \\ \hline \end{matrix} = \begin{matrix} R \\ \left[ \begin{array}{ccc} q_1^T a & q_1^T b & q_1^T c \\ q_2^T a & q_2^T b & q_2^T c \\ q_3^T a & q_3^T b & q_3^T c \end{array} \right] \end{matrix}$$

Pero, por construcción, hay algunos elementos que  
son 0.  $a$  es  $\perp$  a  $q_2$  y  $q_3$ ,  $b$  es  $\perp$  a  $q_3$   
luego

$$R = \begin{bmatrix} q_1^T a & q_1^T b & q_1^T c \\ 0 & q_2^T b & q_2^T c \\ 0 & 0 & q_3^T c \end{bmatrix}$$

Ejemplo: (intentar en casa y comprobar que todo funciona)

$$A = \begin{bmatrix} 1 & 2 & 3 \\ -1 & 0 & -3 \\ 0 & -2 & 3 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{6} & 1/\sqrt{3} \\ -1/\sqrt{2} & 1/\sqrt{6} & 1/\sqrt{3} \\ 0 & -2/\sqrt{6} & 1/\sqrt{3} \end{bmatrix} \begin{bmatrix} \sqrt{2} & \sqrt{2} & \sqrt{18} \\ 0 & \sqrt{6} & -\sqrt{6} \\ 0 & 0 & \sqrt{3} \end{bmatrix} = QR.$$

Esto siempre ocurre

→ Cualquier matriz  $m \times n$   $A$  con columnas independ.

se puede factorizar como  $A = QR$ .

→ La matriz  $Q$  tiene tamaño  $m \times n$  y columnas orthonormales

→ La matriz  $R$  es triang. superior con diagonal positiva.

Factorización QR y mínimos cuadrados

$$\text{Páginas de } Ax = b \rightarrow A^T A \hat{x} = A^T b$$

Si hacemos  $A = QR$ , tenemos que:

$$R^T Q^T Q R \hat{x} = R^T Q^T b$$

$$R^T R \hat{x} = R^T Q^T b$$

$$R \hat{x} = Q^T b \rightarrow \hat{x} = R^{-1} Q^T b$$

Triang.  
sup

back-substitution

Esta parte es muy barata, pero claro, tenemos que hacer Gram-Schmidt, cuyo coste computacional es  $mn^2$  (esto es caro).

## Gram-Schmidt modificado

El algoritmo que hemos desarrollado no es muy estable numéricamente. Desarrollamos otro más estable

Starting from  $\mathbf{a}, \mathbf{b}, \mathbf{c} = \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$  this code will construct  $\mathbf{q}_1$ , then  $\mathbf{B}$ ,  $\mathbf{q}_2$ , then  $\mathbf{C}$ ,  $\mathbf{q}_3$ :

$$\mathbf{q}_1 = \mathbf{a}_1 / \|\mathbf{a}_1\| \quad \mathbf{B} = \mathbf{a}_2 - (\mathbf{q}_1^T \mathbf{a}_2) \mathbf{q}_1 \quad \mathbf{q}_2 = \mathbf{B} / \|\mathbf{B}\|$$

$$\mathbf{C}^* = \mathbf{a}_3 - (\mathbf{q}_1^T \mathbf{a}_3) \mathbf{q}_1 \quad \mathbf{C} = \mathbf{C}^* - (\mathbf{q}_2^T \mathbf{C}^*) \mathbf{q}_2 \quad \mathbf{q}_3 = \mathbf{C} / \|\mathbf{C}\|$$

Equation (11) subtracts **one projection at a time** as in  $\mathbf{C}^*$  and  $\mathbf{C}$ . That change is called **modified Gram-Schmidt**. This code is numerically more stable than equation (8) which subtracts all projections at once.

```
for j = 1:n
    v = A(:,j);
    for i = 1:j-1
        R(i,j) = Q(:,i)'*v;
        v = v - R(i,j)*Q(:,i);
    end
    R(j,j) = norm(v);
    Q(:,j) = v/R(j,j);
end
% modified Gram-Schmidt
% v begins as column j of the original A
% columns q1 to qj-1 are already settled in Q
% compute Rij = q_i^T a_j which is q_i^T v
% subtract the projection (q_i^T v) q_i
% v is now perpendicular to all of q1, ..., qj-1
% the diagonal entries Rjj are lengths
% divide v by its length to get the next qj
% the "for j = 1:n loop" produces all of the qj
```

Obtener QR de forma eficiente

(Linear Alg. Strang)

20/6 - 5/4

Hay dos maneras mejores de obtener QR:

→ Reflexiones de Householder

→ Rotaciones de Givens.

Se basan en algo similar a LU. Recordemos que para construir LU  $\rightarrow EA = U$

↳ hacemos eliminación en A para llegar a U

En este caso  $Q^T A = R$

↳ Multiplicamos A por algo genérico de R tri-g. sup.

# Rotaciones de Givens

(James Wallace Givens, Jr. 1910-1993)

Este es la

matriz de Givens

2-1

$$Q_{21} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

La usamos para hacer un 0 en la posición (2,1)

Use  $Q_{21}$  the way you used  $E_{21}$ , to produce a zero in the (2,1) position. That determines the angle  $\theta$ . Bill Hager gives this example in *Applied Numerical Linear Algebra*:

$$Q_{21}A = \begin{bmatrix} .6 & .8 & 0 \\ -.8 & .6 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 90 & -153 & 114 \\ 120 & -79 & -223 \\ 200 & -40 & 395 \end{bmatrix} = \begin{bmatrix} 150 & -155 & -110 \\ 0 & 75 & -225 \\ 200 & -40 & 395 \end{bmatrix}.$$

The zero came from  $-.8(90) + .6(120)$ . No need to find  $\theta$ , what we needed was  $\cos \theta$ :

$$\cos \theta = \frac{90}{\sqrt{90^2 + 120^2}} \quad \text{and} \quad \sin \theta = \frac{-120}{\sqrt{90^2 + 120^2}}. \quad (2)$$

$$90 \sin \theta + 120 \cos \theta = 0 \rightarrow \text{En realidad, el ángulo}$$

$$\cos \theta = \frac{90}{\sqrt{90^2 + 120^2}} \rightarrow \sin \theta = \frac{-120}{\sqrt{90^2 + 120^2}}$$

$\theta$  da un poco igual

$$(Son \sin \theta y \cos \theta y cuya \sin^2 \theta + \cos^2 \theta = 1)$$

Now we attack the (3,1) entry. The rotation will be in rows and columns 3 and 1. The numbers  $\cos \theta$  and  $\sin \theta$  are determined from 150 and 200, instead of 90 and 120.

$$Q_{31}Q_{21}A = \begin{bmatrix} .6 & 0 & .8 \\ 0 & 1 & 0 \\ -.8 & 0 & .6 \end{bmatrix} \begin{bmatrix} 150 & \cdot & \cdot \\ 0 & \cdot & \cdot \\ 200 & \cdot & \cdot \end{bmatrix} = \begin{bmatrix} 250 & -125 & 250 \\ 0 & 75 & -225 \\ 0 & 100 & 325 \end{bmatrix}.$$

$$\sin \theta = \frac{-200}{\sqrt{150^2 + 200^2}} = -0.8 \quad \cos \theta = \frac{150}{\sqrt{150^2 + 200^2}} = 0.6$$

One more step to  $R$ . The (3, 2) entry has to go. The numbers  $\cos \theta$  and  $\sin \theta$  now come from 75 and 100. The rotation is now in rows and columns 2 and 3:

$$Q_{32}Q_{31}Q_{21}A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & .6 & .8 \\ 0 & -.8 & .6 \end{bmatrix} \left| \begin{array}{ccc} 250 & -125 & \cdot \\ 0 & 75 & \cdot \\ 0 & 100 & \cdot \end{array} \right. = \begin{bmatrix} 250 & -125 & 250 \\ 0 & 125 & 125 \\ 0 & 0 & 375 \end{bmatrix}.$$

We have reached the upper triangular  $R$ . What is  $Q$ ? Move the plane rotations  $Q_{ij}$  to the other side to find  $A = QR$ —just as you moved the elimination matrices  $E_{ij}$  to the other side to find  $A = LU$ :

$$Q_{32}Q_{31}Q_{21}A = R \quad \text{means} \quad A = (Q_{21}^{-1}Q_{31}^{-1}Q_{32}^{-1})R = QR. \quad (3)$$

Por ultimo, como cada  $Q$  es ortogonal  $Q^{-1} = Q^T$

$$A = Q_{21}^T Q_{31}^T Q_{32}^T R = QR$$

\*

\* Producto de matrices ortogonales ( $QQ^T = I$ ) es ortogonal

$$Q_1 Q_1^T = I$$

$$\text{d} Q_1 Q_2 ? (Q_1 Q_2)^T = Q_2^T Q_1^T$$

$$Q_2 Q_2^T = I$$

$$(Q_1 Q_2)^{-1} = Q_2^{-1} Q_1^{-1}$$

$$Q_1 Q_2 (Q_1 Q_2)^{-1} =$$

$$Q_1 Q_2 Q_2^T Q_1^T =$$

$$= Q_1 Q_2 (Q_1 Q_2)^T \checkmark$$

Reflexiones de Householder

Más rápidas que Givens rotations ya que cada una hace 0 todos los elementos bajo la diagonal.  
No entramos en muchos detalles, solo una idea general.

$$A = \begin{bmatrix} 1 & 1 & \dots & 1 \\ a_1 & a_2 & \dots & a_n \\ 1 & 1 & \dots & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} 1 & 1 & \dots & 1 \\ r_1 & r_2 & \dots & r_n \\ 1 & 1 & \dots & 1 \end{bmatrix}$$

Reflexión por  $H_i$

$$H_i = I - 2u_i u_i^T \rightarrow H_i a_i = \begin{bmatrix} \|a_i\| \\ 0 \\ \vdots \\ 0 \end{bmatrix} = r_i$$

$$u_i = \begin{bmatrix} - \\ - \\ - \\ - \\ - \end{bmatrix}$$

(unitario)

$$\begin{bmatrix} \|a_i\| \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} -\|a_i\| \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

→ Cada entrada puede verse como una rotación que permite hacer un 0

De modo que  $H_{n-1} \dots H_1 A = R$ . Además  $H_i$  no solo es ortogonal, sino tb simétrica luego

$$H_i^{-1} = H_i^T = H_i$$

$$H_i = I - 2u_i u_i^T \rightarrow H_i^T = (I - 2u_i u_i^T)^T = I^T - 2(u_i^T)^T u_i^T = I - 2u_i u_i^T$$

$$\begin{aligned} H_i H_i &= (I - 2u_i u_i^T)(I - 2u_i u_i^T) = \\ &= I - 4u_i u_i^T + \underbrace{4u_i u_i^T u_i u_i^T}_{\|u_i\|^2} = I \quad \checkmark \end{aligned}$$

$$A = R \rightarrow \underline{\underline{A = H_1 \dots H_{n-1} R}}$$



Podemos hallar la descomposición QR de una matriz en Python

## numpy.linalg.qr

`linalg.qr(a, mode='reduced')`

Compute the qr factorization of a matrix.

Factor the matrix  $a$  as  $qr$ , where  $q$  is orthonormal and  $r$  is upper-triangular.