reflectores

May 12, 2019

1 Ejercicios de Actualización QR

Clase 15

Equipo 9

```
In [1]: import numpy as np
```

Importamos datos

```
In [2]: x = [x_ for x_ in np.arange(1,10)]
        y = [1.3, 3.5, 4.2, 5.0, 7.0, 8.8, 10.1, 12.5, 13.0]
        n, m = len(x), len(y)
        print(x, ", n =", n)
        print(y, ", m =", m)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9] , n = 9
[1.3, 3.5, 4.2, 5.0, 7.0, 8.8, 10.1, 12.5, 13.0] , m = 9
```

Creamos la matriz A

```
In [3]: A = np.matrix([np.repeat(1,9),
                       x
                      ]).transpose()
        A
```

```
Out[3]: matrix([[1, 1],
                [1, 2],
                [1, 3],
                [1, 4],
                [1, 5],
                [1, 6],
                [1, 7],
                [1, 8],
                [1, 9]])
```

Descomposición vía librería

```
In [4]: q, r = np.linalg.qr(A)
        print("q =", np.round(q, 2), "\nr =", np.round(r, 2))
```

```
q = [[-0.33 -0.52]
 [-0.33 -0.39]
 [-0.33 -0.26]
 [-0.33 -0.13]
 [-0.33  0.  ]
 [-0.33  0.13]
 [-0.33  0.26]
 [-0.33  0.39]
 [-0.33  0.52]]
r = [[ -3.   -15.  ]
 [  0.     7.75]]
```

Multiplicamos el $Q^T y$ para obtener la Q_β

```
In [5]: Q_β = q.T@y
        Q_β
```

```
Out[5]: matrix([[-21.8       ,  11.54149037]])
```

1.0.1 Actualización

Agregamos el valor $x = 10, y = 15.6$ en otra A que denominamos `A_`

```
In [6]: A_ = r.copy()
        A_ = np.append(A_, Q_β.T, axis=1)
        #nuevos valores
        A_ = np.append(A_, np.matrix([1, 10, 15.6]), axis=0)
        A_
```

```
Out[6]: matrix([[ -3.        , -15.        , -21.8       ],
                [  0.        ,   7.74596669,  11.54149037],
                [  1.        ,  10.        ,  15.6       ]])
```

1.0.2 Rotación

Rotación 1 Normas y funciones trigonométricas para la primera columna

```
In [7]: norma = np.linalg.norm([A_[0,0], A_[2,0]], ord=2) #norma eucledіana(A(1,1), A(3,1))
        cos_θ = A_[0,0]/ norma
        sen_θ = A_[2,0] / norma
        sen_θ, cos_θ
```

```
Out[7]: (0.31622776601683794, -0.9486832980505138)
```

Matriz de rotación

```
In [8]: # rot = [cos_teta sen_teta;-sen_teta cos_teta]
        rot = np.matrix([[cos_θ, 0, sen_θ], [0,1,0], [-sen_θ, 0, cos_θ]])
        rot
```

```
Out[8]: matrix([[-0.9486833 ,  0.        ,  0.31622777],
                [ 0.        ,  1.        ,  0.        ],
                [-0.31622777,  0.        , -0.9486833 ]])
```

Matriz con la primera rotación

```
In [9]: A_ = rot@A_
        A_
```

```
Out[9]: matrix([[ 3.16227766, 17.39252713, 25.61444905],
                [ 0.        ,  7.74596669, 11.54149037],
                [ 0.        , -4.74341649, -7.90569415]])
```

Rotación 2 Normas y funciones trigonométricas

```
In [10]: norma = np.linalg.norm([A_[1,1], A_[2,1]], ord=2) #norma eucledіana(A(2,2), A(3,2))
         cos_θ = A_[1,1]/ norma
         sen_θ = A_[2,1] / norma
         cos_θ, sen_θ
```

```
Out[10]: (0.8528028654224418, -0.5222329678670934)
```

Matriz de rotación

```
In [11]: # rot = [cos_teta sen_teta;-sen_teta cos_teta]
         rot = np.matrix([[1, 0, 0], [0, cos_θ, sen_θ], [0, -sen_θ, cos_θ]])
         rot
```

```
Out[11]: matrix([[ 1.        ,  0.        ,  0.        ],
                 [ 0.        ,  0.85280287, -0.52223297],
                 [ 0.        ,  0.52223297,  0.85280287]])
```

Matriz con la segunda rotación

```
In [12]: A_ = rot@A_
         np.round(A_, 2)
```

```
Out[12]: array([[ 3.16, 17.39, 25.61],
                [ 0.  ,  9.08, 13.97],
                [ 0.  , -0.  , -0.71]])
```

1.0.3 Betas

```
In [13]: β_1 = A_[1,2] / A_[1,1]
         β_0 = (A_[0,2] - A_[0,1] * β_1 ) / A_[0,0]
         print("β_0 =", β_0, "\nβ_1 =", β_1)
```

```
β_0 = -0.36
β_1 = 1.538181818181818
```

1.1 Comprobación

Matriz A con los nuevos valores

```
In [14]: A = np.append(A, np.matrix([1,10]), axis=0)
         A
```

```
Out[14]: matrix([[ 1,  1],
                 [ 1,  2],
                 [ 1,  3],
                 [ 1,  4],
                 [ 1,  5],
                 [ 1,  6],
                 [ 1,  7],
                 [ 1,  8],
                 [ 1,  9],
                 [ 1, 10]])
```

y con el nuevo valor

```
In [15]: y.append(15.6)
         y
```

```
Out[15]: [1.3, 3.5, 4.2, 5.0, 7.0, 8.8, 10.1, 12.5, 13.0, 15.6]
```

```
In [16]: A
```

```
Out[16]: matrix([[ 1,  1],
                 [ 1,  2],
                 [ 1,  3],
                 [ 1,  4],
                 [ 1,  5],
                 [ 1,  6],
                 [ 1,  7],
                 [ 1,  8],
                 [ 1,  9],
                 [ 1, 10]])
```

Obtenemos las βs y vemos si coinciden al realizarla manualmente

```
In [17]: q, r = np.linalg.qr(A)

         np.linalg.inv(r).dot(q.T).dot(y)
```

```
Out[17]: matrix([[-0.36       ,  1.53818182]])
```

Los coeficientes coinciden: $-0.36, 1.538$

1.2 Bibliografía

- Notas tomadas en la Clase 15
- Temas vistos en la Clase 15
- Soluciones a ejercicios de actualización vía QR para mínimos cuadrados