EjemploGramSmidth

September 7, 2019

1 Ejemplo de Aplicación Gram Schmidt y QR

En este ejemplo se encuentra la solución del problema $\mathbf{A}\mathbf{x} = \mathbf{b}$ con $\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 2 \\ 1 \\ 5 \\ 4 \end{bmatrix}$.
La solución de minimos cuadrados la podemos encontrar a partir de la descomposicion QR.

Empezamos importando la libreria de Algebra Lineal

```
In [1]: using LinearAlgebra
```

1.1 1. Paso a Paso Gram Schmidt

Empezamos definiendo la matriz **A** y el vector **b**.

```
In [2]: b = [2 1 5 4]' #Transponemos el vector fila para que quede columna
        a1 = ones(4,1) #La primera columna son unos
        a2 = [0;1;1;1] #La segunda columna la definimos usando ;
        a3 = [0 0 1 1]' #Al igual que b, volvemos el vector fila a columna
        A = [a1 a2 a3] #Formamos A haciendo una union de la columnas

Out[2]: 4×3 Array{Float64,2}:
         1.0  0.0  0.0
         1.0  1.0  0.0
         1.0  1.0  1.0
         1.0  1.0  1.0
```

Para la ortogonalizacion empezamos con la primera columna y la volvemos de norma unitaria

```
In [3]: q1 = a1/norm(a1)

Out[3]: 4×1 Array{Float64,2}:
         0.5
         0.5
         0.5
         0.5
```

Continuamos con la segunda columna, le restamos la proyección con respecto a q_1

```
In [4]: q2 = a2-dot(a2,q1)*q1
```

```
Out[4]: 4×1 Array{Float64,2}:
         -0.75
          0.25
          0.25
          0.25
```

Luego, normalizamos q_2

```
In [5]: q2=q2/norm(q2)
```

```
Out[5]: 4×1 Array{Float64,2}:
         -0.8660254037844387
          0.2886751345948129
          0.2886751345948129
          0.2886751345948129
```

Continuamos con la segunda columna, le restamos la proyección con respecto a q_1 y q_2.

```
In [6]: q3 = a3 - dot(a3,q1)*q1 - dot(a3,q2)*q2
```

```
Out[6]: 4×1 Array{Float64,2}:
          1.1102230246251565e-16
         -0.6666666666666667
          0.33333333333333326
          0.33333333333333326
```

normalizamos q_3

```
In [7]: q3 = q3/norm(q3)
```

```
Out[7]: 4×1 Array{Float64,2}:
          1.3597399555105182e-16
         -0.8164965809277261
          0.4082482904638629
          0.4082482904638629
```

Construimos la matriz $\mathbf{Q}$

```
In [8]: Q = [q1 q2 q3]
```

```
Out[8]: 4×3 Array{Float64,2}:
         0.5  -0.866025   1.35974e-16
         0.5   0.288675  -0.816497
         0.5   0.288675   0.408248
         0.5   0.288675   0.408248
```

Calculamos la matriz $\mathbf{R} = \mathbf{Q}^\top \mathbf{A}$.

```
In [9]: R = Q'A
```

```
Out[9]: 3Œ3 Array{Float64,2}:
          2.0            1.5           1.0
          1.11022e-16    0.866025      0.57735
         -2.22045e-16   -3.33067e-16   0.816497
```

Verficiamos que $\mathbf{A} \approx \mathbf{QR}$

```
In [10]: sum(Q*R-A)
```

```
Out[10]: -3.640489776503096e-16
```

Calculamos la solucion, por medio de $\mathbf{Rx} = \mathbf{Q}^\top \mathbf{b}$.

```
In [11]: bp = Q'*b
         x = R\bp
```

```
Out[11]: 3Œ1 Array{Float64,2}:
           1.9999999999999998
          -0.9999999999999997
           3.5
```

1.2 Usando funciones de la libreria de Algebra Lineal

```
In [12]: Q2,R2 = qr(A)
```

```
Out[12]: LinearAlgebra.QRCompactWY{Float64,Array{Float64,2}}
         Q factor:
         4Œ4 LinearAlgebra.QRCompactWYQ{Float64,Array{Float64,2}}:
          -0.5    0.866025    2.22045e-16    9.19739e-17
          -0.5   -0.288675    0.816497      -8.03643e-17
          -0.5   -0.288675   -0.408248      -0.707107
          -0.5   -0.288675   -0.408248       0.707107
         R factor:
         3Œ3 Array{Float64,2}:
          -2.0   -1.5        -1.0
           0.0   -0.866025   -0.57735
           0.0    0.0        -0.816497
```

Observamos que **Q2** es de dimensiones 4x4, usamos solamente sus primeras 3 columnas para realizar el calculo de la solcución.

```
In [13]: x2 = R2\(Q2[:,1:3]'*b)
```

```
Out[13]: 3Œ1 Array{Float64,2}:
           2.000000000000001
          -1.0
           3.4999999999999987
```

```
In [ ]:
```