EXISTENCE & UNIQUENESS · Every $A \in \mathbb{C}^{m\times n}$ $(m \geq n)$ has a full QR factorization ∴ it must have a reduced factorization.

If $A \in \mathbb{C}^{m\times n}$ $(m \geq n)$ is full rank $(r = n) \Longrightarrow$

$$\hat{Q}\hat{R} = A \text{ is unique.}$$

(A) S'pse A is full rank $\Rightarrow$ GS is possible:

$A = \hat{Q}\hat{R}$ and is unique.

(B) S'pse A is not full rank $\Rightarrow$

In the GS algorithm

v_j will be 0 for some j, however we can always add the requisite missing orthonormal q's so that Q is $m \times m$. Since the only requirement made on these additional q_i's is that they be $\perp_1$ this Q can be built in a variety of ways ∴ not unique.

AN APPLICATION OF QR:

S'pse want to solve $Ax=b$, A is square.

Also $b \in col(A)$

$Ax=b \qquad QRx=b$

$Rx=Q^*b$ solve by

$y=Q^*b$ (matrix/vector multiply)

$Rx=y$ (back substitution)

Hence, if QR factorization is available, no need to find A^{-1}

Lecture 8-9

$A \in \mathbb{C}^{m \times n} \qquad A = \begin{bmatrix} | & | & & | \\ a_1 & a_2 & \cdots & a_n \\ | & | & & | \end{bmatrix}$

$$(*) \quad v_j = a_j - (q_1^* a_j) q_1 - (q_2^* a_j) q_2 \cdots - (q_{j-1}^* a_j) q_{j-1}$$

where $q_1 = \frac{a_1}{r_{11}}, \dots, q_n = \frac{a_n - \sum_{i=1}^{n-1} r_{in} q_i}{r_{nn}}$

$$r_{ij} = q_i^* a_j \quad i \neq j \qquad r_{jj} = \left\| a_j - \sum_{i=1}^{j-1} r_{ij} q_i \right\|_2$$

$v_j = P_j a_j$

where $P_j = I - Q_{j-1} Q_{j-1}^*$

$$Q_{j-1} = \underbrace{\begin{bmatrix} | & | & & | \\ q_1 & q_2 & \cdots & q_{j-1} \\ | & | & & | \end{bmatrix}}_{\longleftarrow j-1 \longrightarrow} \Big\updownarrow m$$

is the basis for the following algorithm:

ALGORITHM 7.1
CLASSICAL GRAM-SCHMIDT

```
for j=1:n                      % outer loop
    v_j = a_j                  % each vector has m components
    for i=1:j-1                % inner loop
  (A)   r_ij = q_i^* a_j       % inner product has m "adds", m "mults"
  (B)   v_j = v_j - r_ij q_i   % m "adds" and m "mults"
    end
    r_jj = ||v_j||_2           % m "adds", m "multiplies", 1 sq root
    q_j = v_j / r_jj           % 1 "multiply"
end
```

//

COMPUTATIONAL COMPLEXITY

Gives the estimated resources required to complete a computation via an algorithm.

runtime (wall clock time), operation count (flops) floating point operations, storage (memory)

flops: count # of adds (fast), multiplies (slower), square roots (it depends)

storage: memory (double, single, quadruple precision)

Useful Facts

(1) $$\sum_{l=1}^{n} 1 = n$$

(2) $$\sum_{l=1}^{n} l = \frac{n(n+1)}{2}$$

(3) $$\sum_{l=1}^{n} l^2 = \frac{n(n+1)(2n+1)}{6}$$

Let's estimate the computational complexity of Algorithm 7.1: we'll count flops and not distinguish between adds & multiplies

The outer loop has m ops, repeated n times

$\therefore \sim Cmn$, where C is a constant of order 1.

The inner loop is performed n times $\therefore$

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} 4m \sim \sum_{i=1}^{n} i\, 4m$$

$4m$ is the cost of the A and the B lines.

Use "useful fact" (2) to sum on i:

$$\sum_{i=1}^{n} i\, 4m = \frac{n(n+1)}{2} 4m \approx 2n^2 m$$

So, all told $\sim C_1 mn + C_2 n^2 m$

where C_1 & C_2 are order 1 constants.

- for large m, n $\sim C\, mn^2$, since the second term dominates.
- if $m \sim n$ then $GS \sim Cm^3$

e.g. if $m = 10^2$ $GS \sim 10^6$ flops

if $m = 10^6$ $GS \sim 10^{18}$ flops

very expensive.

THE MODIFIED GRAM-SCHMIDT

It turns out that 7.1 is numerically ill-conditioned (explained later) or "unstable" to numerical errors. The modified GS is better conditioned. Let's go back to

$$(*) \quad v_j = a_j - (q_1^* a_j)q_1 - (q_2^* a_j)q_2 \cdots - (q_{j-1}^* a_j)q_{j-1}$$

where $q_1 = \frac{a_1}{r_{11}}, \ldots, q_n = \frac{a_n - \sum_{i=1}^{n-1} r_{in} q_i}{r_{nn}}$

$$r_{ij} = q_i^* a_j \quad i \neq j \qquad r_{jj} = \left\| a_j - \sum_{i=1}^{j-1} r_{ij} q_i \right\|_2$$

$$q_1 = \frac{P_1 a_1}{\|P_1 a_1\|} \qquad q_2 = \frac{P_2 a_2}{\|P_2 a_2\|}, \quad \cdots \quad q_n = \frac{P_n a_n}{\|P_n a_n\|}.$$

We will write $(*)$ as a matrix vector product:

Using $(*)$ let $v_j = P_j a_j$

Where

$$(\star) \begin{cases} P_j = P_{\perp q_{j-1}} \cdots P_{\perp q_2} P_{\perp q_1} \\ P_1 = I \end{cases}$$

this is the basis for the modified GS

Algorithm 8.1 Modified G-S

for $i=1:n$
  $v_i = a_i$
for $i=1:n$
  $r_{ii} = \|v_i\|_2$
  $q_i = v_i / r_{ii}$
  for $j=i+1:n$
    $r_{ij} = q_i^* v_j$
    $v_j = v_j - r_{ij} q_i$

//

same computational complexity as 7.1

So what are these $P_{\perp q_j}$?

first, we write v_j as obtained by a product of matrixes, applied to a vector:

let $v_j^{\ell=1} = a_j$

$$v_j^{\ell=2} = P_{\perp q_1} v^{\ell=1} = (I - q_1 q_1^*) v_j^{\ell=1}$$

$$v_j^{\ell=3} = P_{\perp q_2} v^{\ell=2}$$

$$\vdots$$

$$v_j \equiv v^{\ell=j} = P_{\perp q_{j-1}} v^{\ell=j-1}$$

where $P_{\perp q_{j-1}} = I - q_{j-1} q_{j-1}^*$

So take the first three of these, in reverse order, to see the pattern:

$$v_j^{(3)} = P_{\perp q_2} v^{(2)} = P_{\perp q_2} P_{\perp q_1} v^{(1)} = P_{\perp q_2} P_{\perp q_1} P_{\perp q_0} a_j$$

where

$$v_j^{(1)} = P_{\perp q_0} a_j = (I - q_0 q_0^*) a_j \qquad q_0 = \begin{bmatrix} \phi \end{bmatrix}$$

So we see how we obtain ☆ above.

What is $P_{\perp q_j}$?

$$P_{\perp q_j} = I - q_j q_j^*$$

Now is $P_j = P_{\perp q_{j-1}} P_{\perp q_{j-2}} \cdots P_{\perp q_1} P_{\perp q_0}$

this last one can be omitted since $= I$.

and $P_j = I - Q_{j-1}\hat{Q}_{j-1}^{*}$?

Because

$$v_j^{(1)} = a_j$$

$$v_j^{(2)} = (I - q_1 q_1^*)\, v_j^{(1)}$$

$$\vdots$$

$$v_j^{(j)} = (I - q_{j-1} q_{j-1}^*)\, v_j^{(j-1)}$$

Take $Q_3 = \begin{bmatrix} | & | & | \\ q_1 & q_2 & q_3 \\ | & | & | \end{bmatrix}$ for example

$$P_4 = I - Q_3 Q_3^* = I - \begin{bmatrix} | & | & | \\ q_1 & q_2 & q_3 \\ | & | & | \end{bmatrix} \begin{bmatrix} - & q_1^* & - \\ - & q_2^* & - \\ - & q_3^* & - \end{bmatrix}$$

$$P_4 = I - \begin{bmatrix} q_1 \end{bmatrix} \begin{bmatrix} - & q_1^* & - \end{bmatrix} - \begin{bmatrix} q_2 \end{bmatrix} \begin{bmatrix} - & q_2^* & - \end{bmatrix} - \begin{bmatrix} q_3 \end{bmatrix} \begin{bmatrix} - & q_3^* & - \end{bmatrix}$$

$$P_4 a_j = (I - q_1 q_1^*)\, a_j - q_2 q_2^* a_j - q_3 q_3^* a_j$$

Now, let's compute

$$P_4 \stackrel{?}{=} P_{\perp q_3} P_{\perp q_2} P_{\perp q_1}$$

$$= (I - q_3 q_3^*)(I - q_2 q_2^*)(I - q_1 q_1^*)$$

$$= (I - q_3 q_3^*)(I - q_2 q_2^* - q_1 q_1^*)$$

$$= I - q_3 q_3^* - q_2 q_2^* - q_1 q_1^*$$

So it agrees. //

Later on we'll discuss why we prefer Algorithm 8.1 over 7.1 and will understand why 8.1 is better conditioned than 7.1

For now, try the HW where you'll need to recreate the lecture 9 results. //