QR and least square

Zecheng Zhang

March 17, 2023

1 QR factorization

We study $A \in \mathbb{R}^{m\times n}$ matrix with linearly independent columns. QR algorithm is a key algorithm in numerical linear algebra. We want to study the column space of A.

Recall the Gram–Schmidt process for producing an orthogonal or an orthonormal basis for any nonzero subspace of $\mathbb{R}^n$. Given a basis $\{x_1, ..., x_p\}$ for a nonzero subspace W, define

$$\begin{aligned}
q_1 &= a_1/r_{11} \\
q_2 &= a_2/r_{22} - \frac{r_{12}}{r_{22}}q_1 \\
q_3 &= a_3/r_{33} - \frac{r_{13}}{r_{33}}q_1 - \frac{r_{23}}{r_{33}}q_2 \\
&\dots \\
q_p &= a_p/r_{pp} - \frac{r_{1p}}{r_{pp}}a_1 - \frac{r_{2p}}{r_{pp}}q_2 - \frac{r_{(p-1)p}}{r_{pp}}q_{p-1},
\end{aligned}$$

where $r_{ij} = q_i^T a_j$ and $r_{jj} = \|a_j - \sum_{i=1}^{j} r_{ij}q_i\|$. Then $\{q_1, ..., q_p\}$ is an orthonormal basis for W, i.e., $span\{a_1, a_2, ..., a_p\} = span\{q_1, q_2, ..., q_p\}$.

Theorem 1.1. If A is an $m \times n$ matrix with linearly independent columns, then A can be factored as $A = QR$, where Q is an $m \times n$ matrix whose columns form an orthonormal basis for Col A and R is an $n \times n$ upper triangular invertible matrix with positive entries on its diagonal.

Proof. Let $a_1, ..., a_n$ be columns of A. Perform Gram-Schmidt, we obtain $Q = [q_1, ..., q_n]$, which is an orthonormal set of vectors whose span is $col(A)$. For a_k, a_k is in $span\{a_1, ..., a_k\} = span\{q_1, ..., q_k\}$. That is there exists $r_{1k}, ..., r_{kk}$ such that $a_k = r_{1k}q_1 + ... + r_{kk}q_k + 0q_{k+1}...0q_n$. Without loss of generality, we assume $r_{kk} > 0$, otherwise multiply r_{kk} and q_k by -1 simultaneously. Denote $Q = [q_1, q_2, ..., q_n]$, $R = [r_1, ..., r_n]$ where $r_k = [r_{1k}, ..., r_{kk}, 0, ..., 0]^t \in \mathbb{R}^n$, recall the matrix multiplication we have $A = QR$. We now claim that R is upper triangular with a positive diagonal (easy to verify) and invertible. Recall $rank(QR) \leq min(rank(Q), rank(R))$. Since $rank(A) = n = rank(Q)$, this implies that $rank(R) = n$. $\square$

When $m > n$, we can append $m - n$ columns to Q to make it a $m \times m$ unitary matrix $\tilde{Q}$. In this process, we will append $m - n$ 0 rows to R to obtain $\tilde{R}$. We call $A = \tilde{Q}\tilde{R}$ full QR of A.

2 Modified QR

The GS-QR algorithm is not numerically stable. For the moment, a stable algorithm is one that is not too sensitive to the effects of rounding off errors. The modified GS is the way to improve

Algorithm 1: Gram Schmidt

Data: $n \geq 0$

for $j = 1$ *to* n **do**
    $v_j = a_j$
    for $i = 1$ *to* $j - 1$ **do**
        $r_{ij} = q_i^t a_j$
        $v_j = v_j - r_{ij} q_i$
    $r_{jj} = \|v_j\|_2$
    $q_j = v_j / r_{jj}$

the stability of the QR algorithm. GS can be expressed as an orthogonal projection:

$$q_1 = \frac{P_1 a_1}{\|P_1 a_1\|}, q_2 = \frac{P_2 a_2}{\|P_2 a_2\|}, ..., q_n = \frac{P_n a_n}{\|P_n a_n\|}, \tag{1}$$

where $P_j \in \mathbb{R}^{m \times m}$ denotes the orthogonal projector onto space spanned by $\{q_1, ...q_{j-1}\}$.

For each j, the GS algorithm computes a single orthogonal projection of rank $m - (j - 1)$, $v_j = P_j a_j$. Recall that: $P_{\perp q}$ denotes the rank $m - 1$ orthogonal projection onto the space orthogonal to q. By the definition of P_j, we can verify (without proof here):

$$P_j = P_{\perp q_{j-1}} ... P_{\perp q_2} P_{\perp q_1}, \tag{2}$$

and $P_{\perp q_1} = I$. As a result,

$$v_j = P_j a_j = P_{\perp q_{j-1}} ... P_{\perp q_2} P_{\perp q_1} a_j. \tag{3}$$

Specifically,

$$\begin{aligned}
v_j^1 &= a_j, \\
v_j^2 &= P_{\perp q_1} v_j^1 = v_j^1 - q_1 q_1^t v_j^1, \\
v_j^3 &= P_{\perp q_2} v_j^2 = v_j^2 - q_2 q_2^t v_j^2, \\
... &\quad ... \\
v_j &= P_{\perp q_{j-1}} v_j^{j-1} = v_j^{j-1} - q_{j-1} q_{j-1}^t v_j^{j-1}.
\end{aligned}$$

We summarize the algorithm in 2.

Algorithm 2: Modified Gram Schmidt

for $i = 1$ *to* n **do**
    $v_i = a_i$

for $i = 1$ *to* n **do**
    $r_{ii} = \|v_i\|$
    $q_i = v_i / r_{ii}$
    for $j = i + 1$ *to* n **do**
        $r_{ij} = q_i^t v_j$
        $v_j = v_j - r_{ij} q_i$

2.1 Operation counts

Each addition, subtraction, multiplication, division and square root counts as one flop. Operation count is the number of flops an algorithm requires.

Theorem 2.1. The Gram-Schmidt algorithm requires $\sim 2mn^2$ flops for a matrix A of size $m \times n$.

Remark 1. The $\sim$ sign here is the asymptotic convergence, i.e.,

$$\lim_{m,n\to\infty} \frac{\text{the total number of flops}}{2mn} = 1. \tag{4}$$

In discussing the operation count, it is standard to discard lower-order terms, since they are usually of little significance unless m and n are small.

Proof. In each i iteration, we have:

1. Line 7: m multiplication and $m-1$ addition.
2. Line 8: m multiplication and m subtraction.

In total we have $\sum_{i=1}^{n}\sum_{j=1}^{n}(4m-1)i \sim 2m^2n$. □

3 Housedolder triangularization

The target of the algorithm is to create a full QR of A. The idea is to applies a sequence of unitary matrices Q_k on the left of A such that, $Q_n...Q_2Q_1A = R$ is upper triangular. Denote $Q = Q_1^tQ_2^t...Q_n^t$, Q is also unitary. This implies that $A = QR$ is a full QR of A. We will discuss how to find Q_i.

3.1 Householder reflector

Each Q_k is chosen to introduce zeros below the diagonal in the $k-$th column while preserving all the zeros previously introduced. In general Q_k operates on rows $k, ..., m$. Each Q_k has the following format:

$$Q_k = \begin{bmatrix} I & 0 \\ 0 & F \end{bmatrix}, \tag{5}$$

where I is the identity matrix of the size $(k-1)\times(k-1)$ and F is unitray of size $(m-k+1)$. Multiplication by F will introduce zeros into $k-$th column. F is called a Householder reflector.

Suppose at the beginning of step k, the entries $k, ..., m$ of $k-$th column are given by the vector $x \in \mathbb{R}^{m-k+1}$. The Householder reflector F should introduce some zeros to x such that $Fx = [\|x\|, 0, ..., 0]^\intercal = \|x\|e_1$. The target now is to construct F such that F will map x to $\|x\|e_1$.

Let us define $v = x - \|x\|e_1$ (please check the picture). By the orthogonal projection formula, we have,

$$Px = (I - \frac{vv^t}{\|v\|^2})x = x - \frac{vv^t}{\|v\|^2}x. \tag{6}$$

This is the orthogonal projection of x onto space which is orthogonal to v. Move twice as far in the same direction; we will have the target vector, i.e.,

$$Fx = x - 2\frac{vv^t}{\|v\|^2}x = (I - 2\frac{vv^t}{\|v\|^2})x. \tag{7}$$

We now derive the Household projector: $F = I - 2\frac{vv^t}{\|v\|^2}$.

Theorem 3.1. F is unitary and Hermitian.

Proof. The proof is straightforward by using the definition. □

We now have $Q_k...Q_1$ which will make A become R. Or we have $Q_k...Q_1A = R$. $Q = Q_1^*...Q_k^*$, but since Q_i are Hermitian, $Q = Q_1...Q_k$.

3.2 The algorithm

Algorithm 3: Householder

Data: $n \geq 0$

for $k = 1$ *to* n **do**
- $x = A_{k:m,k}$
- $v_k = sign(x_1)\|x\|_2 e_1 + x$
- $v_k = v_k/\|v_k\|_2$
- $A_{k:m,k:n} = A_{k:m,k:n} - 2v_k(v_k^t A_{k:m,k:n})$

We can use QR to solve $Ax = b$, where $A \in \mathbb{R}^{n\times n}$ is invertiable. We have $QRx = b$ or $Rx = Q^*b$. This suggests the 3-step method. We now discuss the second step. We will discuss the algorithm

Algorithm 4: QR for $Ax = b$

Data: $n \geq 0$

Compute QR of A;
Compute $y = Q^*x$;
Solve $Rx = y$ for x.

for the last step later.

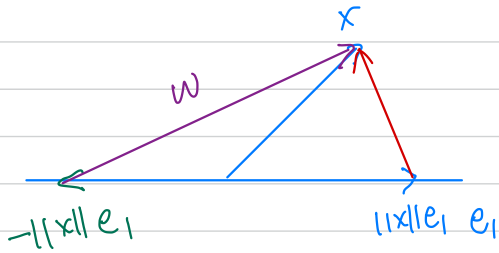
Calculation of Q^*b by a sequence $Q_n...Q_1$ of n operations on b is the same as the operations applied on A to make it triangular. As a result, we have the following algorithm.

Algorithm 5: Compute Q^*b for $Ax = b$

for $k = 1$ *to* n **do**
- $x = A_{k:m,k}$
- $v_k = sign(x_1)\|x\|_2 e_1 + x$
- $v_k = v_k/\|v_k\|_2$
- $b_{k:m} = b_{k:m} - 2v_k v_k^t b_{k:m}$.

The algorithms do not provide us a way to know Q, but by knowing what Q matrix is doing, we can implicitly compute Qx. We know that $Q = Q_1...Q_n$. Q_k will introduce 0 on $k-$

Mar 17 (FRI)



$$Fx = \|x\| e_1$$

$$F = \left(I - \frac{2vv^+}{\|v\|^2}\right) x$$

$$\boxed{-v + x = \|x\| e_1}$$

for stability, $\quad \text{sign}(x_1)\|x\| e_1, \quad x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$

$$Q_k = \begin{pmatrix} I & 0 \\ 0 & F \end{pmatrix}$$

$$\begin{array}{cc} & \begin{array}{cccc} & & k\text{th col} & \end{array} \\ & \\ k\text{th row} \rightarrow & \end{array}$$

$$Q_k \begin{pmatrix} x & x & x & x \\ x & x & x & x \\ 0 & 0 & x & x \\ 0 & 0 & x & x \end{pmatrix} = \begin{pmatrix} x & x & x & x \\ x & x & x & x \\ 0 & 0 & x & x \\ 0 & 0 & \boxed{0} & x \end{pmatrix}$$

$$\underbrace{Q_n Q_{n-1} \cdots Q_1}_{Q^*} A = R$$

$$A = \underbrace{Q_1^* Q_2^* \cdots Q_n^*}_{Q} R$$

pf. $\quad Q_k$ is unitary.

$$F^* F = \left(I - \frac{2vv^*}{\|v\|^2}\right)\left(I - \frac{2vv^*}{\|v\|^2}\right)$$

$$= I - \frac{2vv^*}{\|v\|^2} - \frac{2vv^*}{\|v\|^2} + \frac{4\, vv^* v v^*}{\|v\|^4}$$

$$= I - \cancel{\frac{4 v v^*}{\|v\|^2}} + \cancel{4 \frac{\|v\|^2 v v^*}{\|v\|^4}}$$

$$= I$$

$\Rightarrow$ F is unitary / Hermitian.

$\Rightarrow$ Q_k is unitary / Hermitian $\Rightarrow$ Q is unitary & Hermitian.

Algo 3: Household (will only return "R", we do not have "Q")

$$A_{i:i', j:j'} = \begin{pmatrix} a_{ij} & \cdots & a_{ij'} \\ \vdots & -- & \vdots \\ a_{i'j} & \cdots & a_{i'j'} \end{pmatrix}$$

for $k = 1$ to n

$$x = A_{k:m, k}$$

$$A_{k:m, k} = \begin{pmatrix} a_{kk} \\ \\ a_{mk} \end{pmatrix}$$

$$v_k = -\|x\| e_1 + x$$

$$v_k = \frac{v_k}{\|v_k\|}$$

$$A_{k:m, k:n} = (I - 2 v_k v_k^t) A_{k:m, k:n}$$

$$= A_{k:m, k:n} - 2 v_k v_k^t A_{k:m, k:n}$$

Return R.

Suppose $Ax = b$, $A \in \mathbb{R}^{m \times m}$ invertiable.

want to solve for x

$$QRx = b$$

$$Q^* QRx = Q^* b$$

$$\underline{R}x = Q^* b$$

upper triangular.

$$x = R^{-1} Q^* b$$

Algo 4. QR for $Ax = b$.

1. QR of A (Household)

2. $y = Q^* x$ (?)

   Solve for x

3. $Rx = y$ (talk later)

Calculate $Q^* b$ implicitly w/o having Q^* explicitly.

$$Q^* = Q_n Q_{n-1} \cdots Q_1$$

$$Q_k x = Q_k \begin{pmatrix} x \\ x \\ x \\ x \end{pmatrix} = \begin{pmatrix} x \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Algo 5. compute $Q^* b$ implicitly.

for $k = 1, \dots, n$.

$x = A_{k:m, k}$

$v_k = -\|x\| e_1 + x$

$v_k = \frac{v_k}{\|v_k\|}$

$b_{k:m} = b_{k:m} - 2 v_k v_k^* b_{k:m}$

return $Q^* b$.

Want to calculate Qx implicitly w/o knowing Q.

$Q = Q_1^* Q_2^* \cdots Q_n^*$

$\overset{Q_k \text{ is Hermitian}}{=} Q_1 Q_2 \cdots Q_n$

Q & Q^* have different order, $\Rightarrow$

QA, we will introduce zero from the last column.

Algo 6 Qx implicitly.

for k n back to 1:

$x_{k:m} = x_{k:m} - 2 \underbrace{v_k v_k^*}_{\text{is the same } v_k \text{ as in algo 3.}} x_{k:m}$

Now how can we compute Q?

Method 1.

$$Q = QI = Q[e_1 \cdots e_n]$$

$$= [\underbrace{Qe_1}_{\substack{\text{set } x = e_1 \\ \text{use Algo 6}}}, \dots, \underbrace{Qe_n}_{\substack{\text{set } x = e_n \\ \text{use Algo 6}}}]$$

Method 2.

$$Q^* = Q^* I = [\underbrace{Q^* e_1}_{\text{Algo 5}} \; \cdots \; \underbrace{Q^* e_n}_{\text{Algo 5.}}]$$

To Q, take conjugate of Q^*.

$\boxtimes$

Algorithm 6: Implicit calculation of Qx

for $k = n$ *down to* 1 **do**
  $x_{k:m} = x_{k:m} - 2v_k(v_k^t x_{k:m})$.

column starting from entry $k+1$. This process is implemented by multiplying the vector by the corresponding reflector F_k. We summarize the algorithm as below. The algorithm provides us with one way to compute Q explicitly. We can construct Q by doing QI via Algorithm 6. Specifically, we can compute Qe_1, Qe_2, ..., Qe_n using the algorithm. They are the columns of Q.

Alternatively, we can compute Q^tI via Algorithm 5 and then take transpose or (conjugate if Q^* is comlex) to get Q.

4 Least square

4.1 Motivation

Suppose one has m samples with label y_i, and each sample i has n features $a_{i1}, ..., a_{in}$. We want to approximate y_i by a linear function. More specifically, want to find $x_1, ..., x_n$ such that,

$$\sum_{k=1}^{m}(y_k - \sum_{i=1}^{n} x_i a_{ki})^2.$$

approximation error for k th sample

feature i for sample k

$$\sum_{k=1}^{m} \left(y_k - \sum_{i=1}^{n} x_i a_{ki}\right)^2$$

m samples in total

weight i for ith feature

linear approximation to y_k

If we define the matrix A as

$$a = \begin{bmatrix} a_{11} & ... & a_{1n} \\ a_{21} & ... & a_{2n} \\ ... & ... & ... \\ a_{m1} & ... & a_{mn} \end{bmatrix},$$

and $x = [x_1, ..., x_n]^t$ and $y = [y_1, ..., y_m]^t$, we can reformulate the above minimization problem as

$$\min_{x \in \mathbb{R}^n} \|Ax - y\|^2.$$

4.2 Least square problem

If $A \in \mathbb{R}^{m \times n}$ and b is in $\mathbb{R}^m$, a least-square solution of $A\mathbf{x} = \mathbf{b}$ is an $\hat{\mathbf{x}}$ in $\mathbb{R}^n$ such that

$$||\mathbf{b} - A\hat{\mathbf{x}}|| \leq ||\mathbf{b} - A\mathbf{x}||$$

for all $\mathbf{x}$ in $\mathbb{R}^n$.

Remark 2. Note that Ax is always in the column space of A. As a result, we seek an x such that Ax is the vector in $col(A)$ which is closest to b.

Theorem 4.1. Given A and b as above, let $\hat{b} = \text{proj}_{\text{Col}A} b = Pb$, where P is the orthogonal projector onto range of A. Let $\hat{x}$ in $\mathbb{R}^n$ and it is a least square solution of $Ax = b$ if and only if $\hat{x}$ satisfies $A\hat{x} = \hat{b} = Pb$.

Proof. True. □

4.3 Normal equation

Suppose $\hat{x}$ satisfies $A\hat{x} = \hat{b}$ is the least square solution. We have $b - \hat{b}$ is orthogonal to $col(A)$, it follows that $b - A\hat{x}$ is orthogonal to $col(A)$. We then have

$$a_j^t(b - A\hat{x}) = 0,$$

where a_j is jth column of A. Since a_j^t is the jth row of A^t, we have $A^t(b - A\hat{x}) = 0$. As a result, we have,

$$A^t A x = A^t b.$$

the above equation is called the normal equation for $Ax = b$.

Theorem 4.2. The set of least-squares solutions of $Ax = b$ coincides with the nonempty set of solutions of the normal equations $A^T Ax = A^T b$.

Proof. We have shown that $\hat{x}$ satisfies the normal equation if $\hat{x}$ is the least square solution. Let us prove the converse. Suppose $\hat{x}$ satisfies $A^t A\hat{x} = A^t b$. It follows that $A^t(Ax - b) = 0$, i.e., $Ax - b$ is orthogonal with rows of A^t or columns of A. Consequently, $b = A\hat{x} + (b - A\hat{x})$ is a decomposition of b into sum of a vector in $col(A)$ and $col(A)^\perp$. Due to the uniqueness of the orthogonal projection, $A\hat{x}$ must be the orthogonal projection of b onto $col(A)$. That is $A\hat{x} = \hat{b}$, or $\hat{x}$ is the least square solution. □

Theorem 4.3. Let A be an $m \times n$ matrix. The following statements are logically equivalent:

a. The equation $Ax = b$ has a unique least-squares solution for each b in $\mathbb{R}^m$.

b. The columns of A are linearly independent.

c. The matrix $A^T A$ is invertible.

When these statements are true, the least-squares solution $\hat{x}$ is given by

$$\hat{x} = (A^T A)^{-1} A^T b$$

5 QR

Theorem 5.1. Given an $m \times n$ matrix A with linearly independent columns, let $A = QR$ be a QR factorization of A. Then, for each b in $\mathbb{R}^m$, the equation $Ax = b$ has a unique least-squares solution, given by

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

Proof. Let $\hat{x} = (R)^{-1} Q^T b$. It follows that

$$A\hat{x} = QR\hat{x} = QQ^t b.$$

Recall the POD formulation, $QQ^t b$ is the orthogonal projection of b onto the column space of A, i.e., $QQ^T b = \hat{b}$. This implies $\hat{x}$ is the least square solution. The uniqueness follows from the theorem 4.3. □

6 SVD

Denote the reduced SVD of A as $A = U\Sigma V^T$. Since $range(A) = col(U)$, this suggests that the orthogonal projector $P = UU^t$. It follows that,

$$U\Sigma V^T \hat{x} = UU^t b, \tag{8}$$

or we have,

$$\Sigma V^T \hat{x} = U^t b. \tag{9}$$

We now present the SVD algorithm to compute the least square solution.

Algorithm 7: SVD least square

Compute the reduced SVD of $A = U\Sigma V^T$;
Compute the vector $U^T b$;
Solve the diagonal system $\Sigma w = U^T b$ for w;
Set $x = Vw$.