$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}_{m\times n}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}_{n\times 1}}_{x} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}_{m\times 1}}_{b}$$

CX1104: **Linear Algebra for Computing**

Chap. No : **6.3.3**

Lecture : **Orthogonality**

Topic : **Gram–Schmidt Process**

Concept : **Using Matlab to get QR and 4 cases of A**

Instructor: **A/P Chng Eng Siong**

TAs: **Zhang Su, Vishal Choudhari**

Last updated: 19 Sep 2021

Rev: 29th June 2020

Using Matlab for QR

We will consider only the two common cases:

i) A is a square matrix
ii) A has dimension mxn (m>n)

Using Matlab, there are 2 options, complete (full) vs economy decomposition.

Matlab

```
qr     Orthogonal-triangular decomposition.
   [Q,R] = qr(A), where A is m-by-n, produces an m-by-n upper triangular
   matrix R and an m-by-m unitary matrix Q so that A = Q*R.

   [Q,R] = qr(A,0) produces the "economy size" decomposition.
   If m>n, only the first n columns of Q and the first n rows of R are
   computed. If m<=n, this is the same as [Q,R] = qr(A).
```

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.

Parameters: **a** : *array_like, shape (M, N)*
Matrix to be factored.

mode : *{'reduced', 'complete', 'r', 'raw'}, optional*
If K = min(M, N), then
- 'reduced' : returns q, r with dimensions (M, K), (K, N) (default)
- 'complete' : returns q, r with dimensions (M, M), (M, N)

Note: option for numpy 'reduced' == matlab 'economy' when performing QR

Ref: 1) https://en.wikipedia.org/wiki/QR_decomposition
2) http://ee263.stanford.edu/lectures/qr.pdf

QR using Matlab and the 4 cases

We explore QR of A for the following 4 cases

1) Square A matrix (size 3x3)
   - Ex1) where A has 3 independent col
   - Ex2) where A has 2 independent col, and 1 dependent col

2) Tall A matrix (size 3x2)
   - Ex3) where A has 2 independent col
   - Ex4) where A col are dependent (col2 == 2xcol1)

Study the effect decomposition has on Q and R, as well
as selection of Matlab economy vs complete (full) QR decomposition.

Introducing $A = QR$ for Square A

Example 1: A is 3x3 square and has independent column

Square matrix [edit]

Any real square matrix A may be decomposed as

$$A = QR,$$

where Q is an orthogonal matrix (its columns are orthogonal unit vectors meaning $Q^\mathsf{T} = Q^{-1}$) and R is an upper triangular matrix (also called right triangular matrix). If A is invertible, then the factorization is unique if we require the diagonal elements of R to be positive.

If instead A is a complex square matrix, then there is a decomposition $A = QR$ where Q is a unitary matrix (so $Q^* = Q^{-1}$).

If A has n linearly independent columns, then the first n columns of Q form an orthonormal basis for the column space of A. More generally, the first k columns of Q form an orthonormal basis for the span of the first k columns of A for any $1 \le k \le n$.[1] The fact that any column k of A only depends on the first k columns of Q is responsible for the triangular form of R.[1]

```
>> A = [1 2 3; 1 2 1; 1 1 3]

A =

     1     2     3
     1     2     1
     1     1     3

>> rank(A)

ans =

     3
```

```
>> [Q,R] = qr(A)

Q =

   -0.5774   -0.4082   -0.7071
   -0.5774   -0.4082    0.7071
   -0.5774    0.8165   -0.0000


R =

   -1.7321   -2.8868   -4.0415
         0   -0.8165    0.8165
         0         0   -1.4142
```

```
>> rank(R)

ans =

     3
```

```
>> Q*R

ans =

    1.0000    2.0000    3.0000
    1.0000    2.0000    1.0000
    1.0000    1.0000    3.0000
```

```
>> Q'*Q

ans =

    1.0000    0.0000    0.0000
    0.0000    1.0000   -0.0000
    0.0000   -0.0000    1.0000

>> Q*Q'

ans =

    1.0000   -0.0000   -0.0000
   -0.0000    1.0000   -0.0000
   -0.0000   -0.0000    1.0000
```

Matlab Notation:

$$Q' == Q^T$$

When A is square and has independent column, factorizing A to QR, then
Q'*Q == Q*Q' == identity matrix
And R which is square is invertible!

Square A with dependent col

Example 2: A is 3x3 square and
A has 1 dependent column

```
>> A = [1 2 3; 1 2 3; 1 1 2]

A =

     1     2     3
     1     2     3
     1     1     2

>> rank(A)

ans =

     2
```

```
>> [Q,R] = qr(A)

Q =

   -0.5774   -0.4082   -0.7071
   -0.5774   -0.4082    0.7071
   -0.5774    0.8165   -0.0000


R =

   -1.7321   -2.8868   -4.6188
         0   -0.8165   -0.8165
         0         0   -0.0000
```

```
>> Q*R

ans =

    1.0000    2.0000    3.0000
    1.0000    2.0000    3.0000
    1.0000    1.0000    2.0000
```

```
>> rank(R)

ans =

     2
```

```
>> Q'*Q

ans =

    1.0000    0.0000    0.0000
    0.0000    1.0000   -0.0000
    0.0000   -0.0000    1.0000

>> Q*Q'

ans =

    1.0000   -0.0000   -0.0000
   -0.0000    1.0000   -0.0000
   -0.0000   -0.0000    1.0000
```

When A is square and has dependent column,
It still can be decomposed into QR, and
Q'*Q == Q*Q' == identity matrix
BUT now, R which is square is NOT invertible!
R has rank 2 bcos A has 2 independent col.

Ref: https://en.wikipedia.org/wiki/QR_decomposition

Introducing $A = QR$ for Tall A $(m>n)$

Rectangular matrix [edit]

More generally, we can factor a complex $m \times n$ matrix A, with $m \geq n$, as the product of an $m \times m$ unitary matrix Q and an $m \times n$ upper triangular matrix R. As the bottom $(m-n)$ rows of an $m \times n$ upper triangular matrix consist entirely of zeroes, it is often useful to partition R, or both R and Q:

$$A = QR = Q\begin{bmatrix} R_1 \\ 0 \end{bmatrix} = [Q_1 \quad Q_2]\begin{bmatrix} R_1 \\ 0 \end{bmatrix} = Q_1 R_1,$$

where R_1 is an $n \times n$ upper triangular matrix, 0 is an $(m - n) \times n$ zero matrix, Q_1 is $m \times n$, Q_2 is $m \times (m - n)$, and Q_1 and Q_2 both have orthogonal columns.

Golub & Van Loan (1996, §5.2) call Q_1R_1 the *thin QR factorization* of A; Trefethen and Bau call this the *reduced QR factorization*.[1] If A is of full rank n and we require that the diagonal elements of R_1 are positive then R_1 and Q_1 are unique, but in general Q_2 is not. R_1 is then equal to the upper triangular factor of the Cholesky decomposition of $A^* A$ (= A^TA if A is real).

Note:

$col(Q_1)$== col(A),

while

$col(Q_2)$ = orthogonal complement of col(A)

see example 3 and 4 (in next pages)

Tall A

$$A = QR = Q\begin{bmatrix} R_1 \\ 0 \end{bmatrix} = [Q_1 \quad Q_2]\begin{bmatrix} R_1 \\ 0 \end{bmatrix} = Q_1 R_1,$$

```
A =

     1     2
     1     2
     1     1

>> [Q,R] = qr(A)

Q =

   -0.5774   -0.4082   -0.7071
   -0.5774   -0.4082    0.7071
   -0.5774    0.8165   -0.0000


R =

   -1.7321   -2.8868
         0   -0.8165
         0         0
```

```
>> Q'*Q

ans =

    1.0000    0.0000    0.0000
    0.0000    1.0000   -0.0000
    0.0000   -0.0000    1.0000

>> Q*Q'

ans =

    1.0000   -0.0000   -0.0000
   -0.0000    1.0000   -0.0000
   -0.0000   -0.0000    1.0000
```

[Q,R] = qr(A) (complete decomposition)

- Complete Q has size == 3x3 (and it an orthogonal matrix even though A is 3x2.
- R is 3x2 (row 1 and 2 of R are non-zero bcos A has 2 independent col)

Q_1 is the first 2 columns of Q (bcos A has 2 independent col)

$\mathrm{col}(Q_1) == \mathrm{col}(\mathrm{A})$

Q_2 is the last column of Q

Example 3: A is tall (3x2) and A has 2 independent column

```
A =

     1     2
     1     2
     1     1

>> [Q,R] = qr(A,0)

Q =

   -0.5774   -0.4082
   -0.5774   -0.4082
   -0.5774    0.8165


R =

   -1.7321   -2.8868
         0   -0.8165
```

```
>> Q'*Q

ans =

    1.0000    0.0000
    0.0000    1.0000

>> Q*Q'

ans =

    0.5000    0.5000   -0.0000
    0.5000    0.5000    0.0000
   -0.0000    0.0000    1.0000
```

[Q,R] = qr(A,0) (using economy decomposition)

reduced Q has size == 3x2.

Note that Q'*Q = I (size (2x2))

While Q*Q' = P

a projection matrix size 3x3.

A projection matrix: see Strang lecture 16

https://www.youtube.com/watch?v=osh80YCg_GM

Tall A

$$A = QR = Q\begin{bmatrix} R_1 \\ 0 \end{bmatrix} = [Q_1 \quad Q_2]\begin{bmatrix} R_1 \\ 0 \end{bmatrix} = Q_1 R_1,$$

Example 4: A is tall (3x2) and A has dependent column

```
A =

     1     2
     1     2
     1     2

>> [Q,R] = qr(A)

Q =

   -0.5774    0.8165   -0.0000
   -0.5774   -0.4082   -0.7071
   -0.5774   -0.4082    0.7071


R =

   -1.7321   -3.4641
         0   -0.0000
         0         0
```

```
>> Q*R

ans =

    1.0000    2.0000
    1.0000    2.0000
    1.0000    2.0000
```

[Q,R] = qr(A) (complete decomposition)
Complete Q has size == 3x3 even though A is 3x2
And R is 3x2 (Only row 1 of R is non-zero, bcos there is ONLY 1 independent col in A)

Q_1 is ONLY the first column of Q (bcos A has ONLY 1 independent col), $col(Q_1) == col(A)$
Q_2 is the last 2 columns of Q

```
A =

     1     2
     1     2
     1     2

>> [Q,R] = qr(A,0)

Q =

   -0.5774    0.8165
   -0.5774   -0.4082
   -0.5774   -0.4082


R =

   -1.7321   -3.4641
         0   -0.0000
```

```
>> Q*R

ans =

    1.0000    2.0000
    1.0000    2.0000
    1.0000    2.0000
```

[Q,R] = qr(A,0) (economy decomposition)
reduced Q has size == 3x2