Linear Algebra

Orthogonal Matrices and QR Decomposition

Automotive Intelligence Lab.

HANYANG UNIVERSITY

AI LAB Automotive Intelligence

Contents

QR Decomposition

Definition of QR Decomposition

- **Decompose matrix with $\boxed{\text{standard orthogonal basis.}}$ vector which is found using Gram-Schmidt.**

- **Matrix $\boldsymbol{Q}$**
  - Set of $\boxed{\text{standard orthogonal basis.}}$ $q_1, \cdots, q_n$ obtained through the Gram-Schmidt
  - $\boldsymbol{Q}$ is obviously different from the original matrix. $A \neq Q$
    - Assuming original matrix was not orthogonal.
    - Lost information about that matrix.

- **Fortunately, lost information can be retrieved and stored in another matrix $\boldsymbol{R}$.**
  - $\boldsymbol{R}$ multiplies to $\boldsymbol{Q}$.
  - Then…, how to create $\boldsymbol{R}$?

$$\begin{tikzcd} A = \arrow[r, bend left] & Q \cdot \arrow[r, bend left] & R \end{tikzcd}$$

Creating R

- **Comes right from the definition of QR.**

$Q^{-1} = Q^T.$

$$A = QR$$

$$Q^T A = Q^T Q R$$

$$\boxed{Q^T A} = R$$

Definition of QR

- **Advantage of orthogonal matrices that can be seen from the above Definition.**
  - Solve matrix equations **without** having to worry about **computing the inverse**.

$R = Q^T \cdot A.$

$$\overset{A}{\begin{bmatrix} | & | & & | \\ a_1 & a_2 & \cdots & a_n \\ | & | & & | \end{bmatrix}} = \overset{Q}{\begin{bmatrix} | & | & & | \\ q_1 & q_2 & \cdots & q_n \\ | & | & & | \end{bmatrix}} \begin{bmatrix} a_1 \cdot q_1 & a_2 \cdot q_1 & \cdots & a_n \cdot q_1 \\ a_1 \cdot q_2 & a_2 \cdot q_2 & \cdots & a_n \cdot q_2 \\ \vdots & \vdots & \ddots & \vdots \\ a_1 \cdot q_n & a_2 \cdot q_n & \cdots & a_n \cdot q_n \end{bmatrix}$$

Overall form of QR decomposition

Simplification of QR Decomposition

- **Consider $\boldsymbol{a_1 \cdot q_2}$**
  - $a_1 \cdot q_2 = 0$ because a_1 is orthogonal to q_2 .
- **For $\underline{\boldsymbol{a_i \cdot q_j, i < j}}$**
  - $\underline{a_i \cdot q_j = 0}$
    - Because a_i is orthogonal to q_j for $\boldsymbol{i < j}$.

$$\underline{\boldsymbol{q_1 = a_1}}$$

0.

$$\boldsymbol{q_2 = \circled{a_2} - proj_{q_1}(a_2)}$$

$$\boldsymbol{\circled{q_3} = a_3 - proj_{q_1}(a_3) - proj_{q_2}(a_3)}$$

$$\vdots$$

$$\boldsymbol{q_k = a_k - \sum_{j=1}^{k-1} proj_{q_j}(a_k)}$$

$$a_2 = q_2 + \underline{proj_{q_1}(a_2)}.$$

q_1과 같은 방향.

$$q_3 \cdot a_2 = q_3 (q_2 + proj_{q_1}(a_2))$$

$$= \begin{bmatrix} | & | & & | \\ q_1 & q_2 & \cdots & q_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} a_1 \cdot q_1 & a_2 \cdot q_1 & \cdots & a_n \cdot q_1 \\ 0 & a_2 \cdot q_2 & \cdots & a_n \cdot q_2 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_n \cdot q_n \end{bmatrix}$$

0.

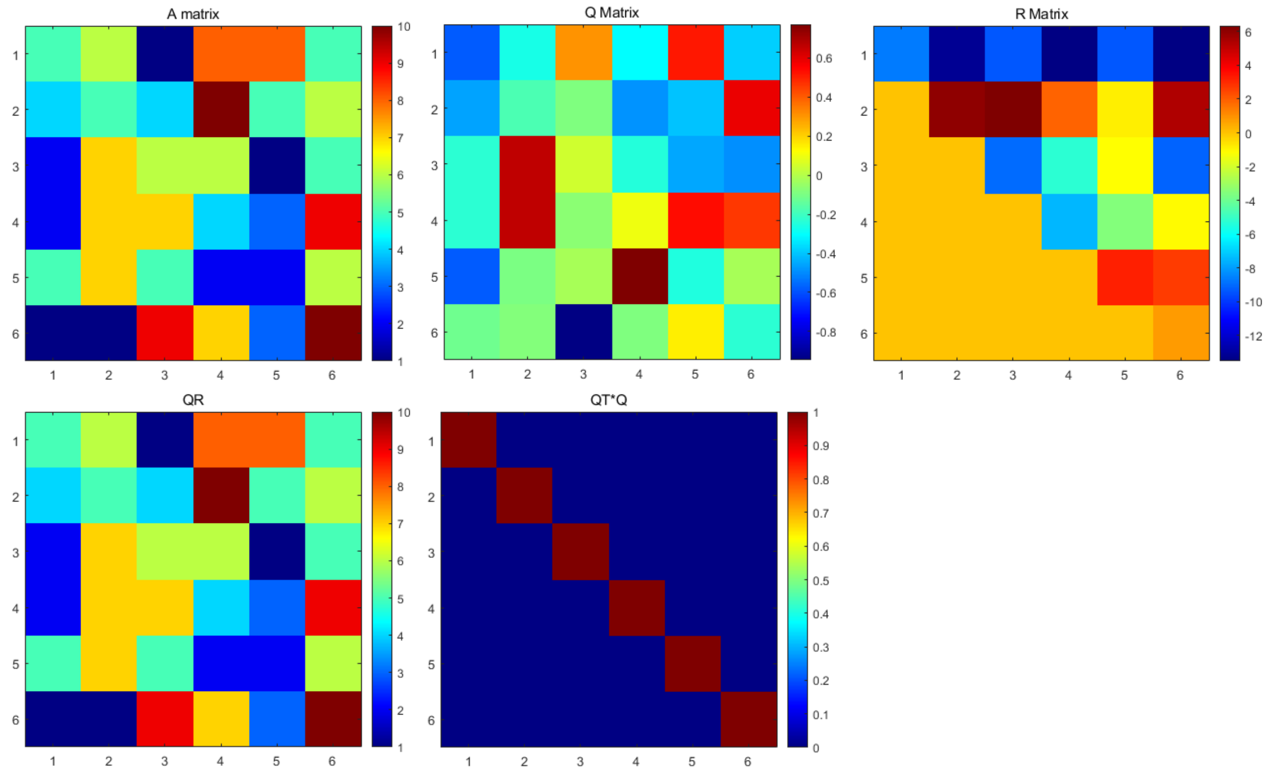
Simplification of QR decomposition

Features of QR Decomposition

- $A = QR$
  - $A - QR$ is zeros matrix.
- **Q times its transpose gives the identity matrix.**
- **R matrix: Always Upper triangular**
  - It will be explained in the next section.

```
% Clear workspace, command window, and close
all figures
clc; clear; close all;

% Random integer matrix A
A = randi(10, 6);

% QR decomposition
[Q,R] = qr(A);

% Visualize the results
figure;
imagesc(A); % Display the matrix as a color
image
title('A matrix');
colorbar; % Show a color scale
colormap jet; % Use the jet color map
axis equal tight;  % Adjust axes to fit the
data

figure;
imagesc(Q);
title('Q Matrix');
colorbar;
colormap jet;
axis equal tight;

figure;
imagesc(R);
title('R Matrix');
colorbar;
colormap jet;
axis equal tight;

figure;
imagesc(Q*R);
title('QR');
colorbar;
colormap jet;
axis equal tight;

figure;
imagesc(Q' * Q);
title('QT*Q');
colorbar;
colormap jet;
axis equal tight;
```

MATLAB code



QR decomposition of a random numbers matrix

Sizes of Q and R

- **Depend on size of the to-be-decomposed matrix A**
- **Whether QR decomposition is Economy or Full.**
  - **Economy** called reduced.
  - **Full** called complete.

Overview of All Possible Sizes of Q and R

- Fig 1. shows an overview of all possible sizes.
- "?" indicates that the matrix elements depend on values in A.
  - Not identity matrix

	A	Q	Q^TQ	QQ^T	R
Square full-rank	$M \times M$ $r = M$	$M \times M$ $r = M$	I_M	I_M	$M \times M$ $r = M$
Square singular	$M \times M$ $r = K < M$	$M \times M$ $r = M$	I_M	I_M	$M \times M$ $r = k$
Tall "full"	$M > N$ $r = K$	$M \times M$ $r = M$	I_M	I_M	$M \times M$ $r = k$
Tall "economy"	$M > N$ $r = K$	$M \times N$ $r = N$	I_N	?	$M \times N$ $r = K$
Wide	$M < N$ $r = K$	$M \times M$ $r = M$	I_M	I_M	$M \times N$ $r = K$

Fig 1. Sizes of Q and R depending on size of A

Code Exercise of Orthogonal Matrix using MATLAB

- Notice optional second input ‘complete’, which produces a full QR decomposition.
- Setting that to ‘reduced’, gives economy-mode QR decomposition, in which Q is same size as A.

```matlab
% Clear workspace, command window, and close all figures
clc; clear; close all;

% Matrix A
A = [1; -1];
[Q,R] = qr(A); % Full QR decomposition
[Q_econ,R_econ] = qr(A, "econ"); % Economy-mode QR decomposition, Q is smae size as matrix A

% Scale to make integer matrix
Q = Q*sqrt(2);
Q_econ = Q_econ*sqrt(2);

% Display the results
disp("Q")
disp(Q);
disp("Q_econ")
disp(Q_econ);
```

MATLAB code of orthogonal matrix

Rank of Orthogonal Matrix

- **Rank of $\boldsymbol{Q}$ is always maximum possible rank.**
  - It is possible to craft more than $M > N$ orthogonal vectors from a matrix with N columns.

- **Rank of $\boldsymbol{Q}$**
  - $\boldsymbol{M}$ for all square $\boldsymbol{Q}$ matrices
  - $\boldsymbol{N}$ for economy $\boldsymbol{Q}$ matrices

- **Rank of $\boldsymbol{R}$**
  - Same as rank of $\boldsymbol{A}$

- **Difference in rank between $\boldsymbol{Q}$ and $\boldsymbol{A}$ resulting from orthogonalization**
  - Q spans all of $\mathbb{R}^M$ even if the column space of $\boldsymbol{A}$ is only lower-dimensional subspace of $\mathbb{R}^M$.
    - Important reason why the singular value decomposition is so useful for revealing properties of a matrix, including its rank and null space.
  - Another reason to look forward to learning about SVD in Chapter 14!

Property of QR Decomposition

- **QR decomposition is not unique for all matrix sizes and ranks.**
  - It is possible to obtain $\boldsymbol{A} = \boldsymbol{Q}_1\boldsymbol{R}_1$ and $\boldsymbol{A} = \boldsymbol{Q}_2\boldsymbol{R}_2$ where $\boldsymbol{Q}_1 \neq \boldsymbol{Q}_2$.

- **All QR decomposition results have the same properties described in this section.**

- **QR decomposition can be made unique when given additional constraints.**
    - E.g., Positive values on diagonals of $\boldsymbol{R}$
  - But! **Not necessary** in most cases.
    - Not implemented in MATLAB.

Orthogonalization

- **Orthogonalization works column-wise from left to right.**
  - Later columns in $\boldsymbol{Q}$ are orthogonalized to earlier columns of $\boldsymbol{A}$.
- **Lower triangle of $\boldsymbol{R}$ comes from** $\boxed{\text{orthogonalized pairs of vectors}}$**.**
- **Earlier columns in $\boldsymbol{Q}$ are not orthogonalized to later columns of $\boldsymbol{A}$.**
  - Not expect their dot products to be zero.

- **Columns $\boldsymbol{i}$ and $\boldsymbol{j}$ of $\boldsymbol{A}$ were already orthogonal.**
  - Corresponding $(i, j)^{th}$ element in $\boldsymbol{R}$ would be $\boxed{\text{zero}}$.
- **If compute QR decomposition of orthogonal matrix,**
  - $\boldsymbol{R}$ will be $\boxed{\text{diagonal}}$ matrix.
    - Norms of each column in $\boldsymbol{A}$.
- **If $\boldsymbol{A} = \boldsymbol{Q}$, $\boldsymbol{R}$ is same as** $\boxed{I}$**.**
  - Comes from equation solved for $\boldsymbol{R}$.

QR and Inverses

- **More numerically stable way to compute matrix inverse**
  - When using QR decomposition.
- **Writing out QR decomposition formula and inverting both sides of equation.**
  - Apply the **LIVE EVIL** rule as we learned before.
- **Inverse of A**
  - Same as $\boxed{\text{inverse}}$ of R times $\boxed{\text{transpose.}}$ of Q.
  - Q is numerically stable.
    - Due to **Householder reflection algorithm**.
  - R is numerically stable.
    - Due to results from **matrix multiplication**.
- **Need to invert R explicitly.**
  - **Inverting triangular matrices** is highly numerically stable.
    - Through $\boxed{\text{back substitution}}$.

$$A = QR$$

$$A^{-1} = (QR)^{-1}$$

$$A^{-1} = R^{-1}Q^{-1}$$

$$A^{-1} = R^{-1}Q^{T}$$

Compute matrix inverse using QR decomposition

Key Point of QR Decomposition

- **Provide more numerically stable way to invert matrices.**
  - Compared to algorithm presented in previous lecture.
- **On the other hand, some matrices are still very difficult to invert.**
  - Theoretically invertible but are close to singular.
- **QR decomposition doesn't guarantee high-quality inverse.**
  - Rotten apple dipped in honey is still rotten…!

Summary

Summary

- **Orthogonal matrix**
  - All columns are pair-wise orthogonal and $\text{norm} = 1$.
  - Key to several matrix decompositions.
    - QR, eigen, singular value decomposition.
  - Important in geometry and computer graphics.
    - E.g. pure rotation matrices.
- **Can transform a nonorthogonal matrix into an orthogonal matrix.**
  - Via Gram-Schmidt procedure.
  - Involves applying orthogonal vector decomposition.
    - To isolate the component of each column.
    - Each column is orthogonal to all previous columns, previous meaning left to right.
- **QR decomposition is the result of Gram-Schmidt.**
  - Technically, it is implemented by a more stable algorithm.
  - But GS is still the right way to understand it.

Code Exercises

Characteristic of matrix Q

- A square Q has the following equalities:

$$Q^TQ = QQ^T = Q^{-1}Q = QQ^{-1} = I$$

- Demonstrate this in code by computing Q from a random-numbers matrix, then compute Q^T and Q^{-1}. Then show that all four expressions produce the identity matrix.

```
% Generate a 5x5 random matrix and compute the QR decomposition


random_matrix = randn(5, 5);
% Generate Q matrix
Q =
% Get Transpose of Q & Inverse of Q
Qt =
Qi =

% disp QTQ, QQT, QIQ, QQI
```

Sample code

Full, Economy Sized matrix Q and Its Inverse

- This exercise will highlight one feature of the R matrix that is relevant for under-standing how to use QR to implement least squares (lecture 12): when A is tall and full column-rank, the first N rows of R are upper-triangular, whereas rows $N+1$ through M are zeros. Confirm this in MATLAB using a random 10×4 matrix. Make sure to use the complete (full) QR decomposition, not the economy (compact) decomposition.

- Of course, R is noninvertible because it is nonsquare. But (1) the submatrix comprising the first N row is square and full-rank (when A us full column-rank) and thus has a full inverse, and (2) the tall R has a pseudoinverse. Compute both inverses, and confirm that the full inverse of the first N rows of R equals the first N columns of the pseudoinverse of the tall R.

```
% Create a random 10x4 matrix
A = randn(10, 4);

% Compute the complete QR decomposition
% economy sized R
[~, R] = ;
% full sized R
[~, fullR] = ;

% Examine R (rounded to 3 decimal places)
disp('R:');
disp(round(R, 3));
disp('fullR:');
disp(round(fullR, 3));
```

```
% Invertible submatrix (first 4x4 part of R)
Rsub = ;


% Inverses
% calculate full inverse of Rsub
Rsub_inv = ;
% calculate left inverse of R
Rleftinv = ;


% Display both inverses
disp('Full inverse of R submatrix:');
disp(round(Rsub_inv, 3));


disp('Left inverse of R:');
disp(round(Rleftinv, 3));
```

Sample code

THANK YOU FOR YOUR ATTENTION

THANK YOU
FOR YOUR ATTENTION