Nicole Cruz
CSCI 166
Levinson
Dec 9 2022

QR Decomposition

Introduction

During Numerical Analysis, we looked at ways to decompose a matrix, one such way being LU Decomposition. However, this is not the only form of decomposition we are able to investigate. QR decomposition introduces a new way to decompose a matrix into two matrices: Q & R. Where LU decomposition was introduced only for square matrices, we can expand the decompositions to that of rectangular matrices, specifically, QR decomposition works for both square and rectangular matrices.

Theory

Suppose we have such a matrix A of dimensions N x M. Through QR decomposition, we can decompose A into two matrices, Q and R with dimensions N x M and M x M respectively, where Q is a matrix such that its columns are *orthonormal*, and R is *upper-triangular* with strictly positive diagonal entries.

Recall, orthogonal means that the inner product of vectors $< \vec{a}, \vec{b} > = 0$, meaning vectors $\vec{a}$ & $\vec{b}$ in space are taken at right angles from one another.

From linear algebra, we recall that orthonormal means that both orthogonal and a 2-norm of 1. Recall that we define 2-norm (also referred to as the Euclidean distance) as:

$$||\vec{x}||_2 = \sqrt{x_1^2 + x_2^2 + x_3^2 + \ \dots \ + x_n^2} \text{ for some } \vec{x} = [x_1, x_2, \dots, x_n]^T$$

In order to receive the columns of Q, we utilize the Gram-Schmidt process. The Gram-Schmidt process takes a basis of vectors and makes them all mutually orthogonal.

Def: *(Gram-Schmidt)* Given linearly independent vectors $\{\vec{a}_1, \vec{a}_2, \dots, \vec{a}_n\}$, then we can redefine a new set of vectors $\{\vec{q}_1, \vec{q}_2, \dots, \vec{q}_m\}$ which are mutually orthogonal.

In previous classes, we saw the formula for polynomials:

$$p_k = x^k - \sum_{j=0}^{k-1} \frac{<p_j, x^k>}{<p_j, p_j>}$$

In a similar format for vectors, where each vector is a *column* of their respective matrix, we have the following formula:

$$\vec{q}_1 = \frac{\vec{a}_1}{||\vec{a}_1||_2}$$

$$\vec{q}_2 = \frac{\vec{a}_2 - \frac{<\vec{a}_2, \vec{q}_1>}{<\vec{q}_1, \vec{q}_1>} \vec{q}_1}{||(\vec{a}_2 - \frac{<\vec{a}_2, \vec{q}_1>}{<\vec{q}_1, \vec{q}_1>} \vec{q}_1)||_2}$$

…

$$\vec{q}_m = \frac{\vec{a}_m - \frac{<\vec{a}_m, \vec{q}_1>}{<\vec{q}_1, \vec{q}_1>} \vec{q}_1 - \frac{<\vec{a}_m, \vec{q}_2>}{<\vec{q}_2, \vec{q}_2>} \vec{q}_2 - \dots - \frac{<\vec{a}_m, \vec{q}_{m-1}>}{<\vec{q}_{m-1}, \vec{q}_{m-1}>} \vec{q}_{m-1}}{||\vec{a}_m - \frac{<\vec{a}_m, \vec{q}_1>}{<\vec{q}_1, \vec{q}_1>} \vec{q}_1 - \frac{<\vec{a}_m, \vec{q}_2>}{<\vec{q}_2, \vec{q}_2>} \vec{q}_2 - \dots - \frac{<\vec{a}_m, \vec{q}_{m-1}>}{<\vec{q}_{m-1}, \vec{q}_{m-1}>} \vec{q}_{m-1}||_2}$$

$$\vec{q}_m = \vec{a}_m - \frac{<\vec{a}_m, \vec{q}_1>}{<\vec{q}_1, \vec{q}_1>}\vec{q}_1 - \frac{<\vec{a}_m, \vec{q}_2>}{<\vec{q}_2, \vec{q}_2>}\vec{q}_2 - \ \ldots \ - \frac{<\vec{a}_m, \vec{q}_{m-1}>}{<\vec{q}_{m-1}, \vec{q}_{m-1}>}\vec{q}_{m-1}$$

$$\text{where each } <\vec{a}_i, \vec{b}_j> \ = a_1b_1 + a_2b_2 + \ \ldots \ + a_ib_j$$

Since we are focusing on orthonormal columns, then

$$<\vec{q}_i, \vec{q}_j> \ = ||\vec{x}||_2 = 1 \text{ when i = j}$$

Note: Each inner product we see within the equation is denoted as the dot product between the two given vectors.

If familiar with linear algebra, we are calculating the projection of $\vec{a}_{k+1}$ onto the span of the columns of Q:

$$\vec{q}_k = \vec{a}_{k+1} - Proj_{span\{\vec{q}_1, \ldots, \vec{q}_k\}}(\vec{a}_{k+1})$$

Thus, we also show that the span of orthogonal vectors found is equal to the span of the original linearly independent vectors.

Thus, Q (N x M) is:

$$Q = [\ \vec{q}_1, \vec{q}_2, \ldots, \vec{q}_m\ ]$$

To find the R matrix, which we stated earlier is upper-triangular, we take the coefficients of each projection such that R (M x M) resembles:

$<\vec{a}_1, \vec{q}_1>$	…	$<\vec{a}_m, \vec{q}_1>$
0	$<\vec{a}_i, \vec{q}_j>$ where i = j	$<\vec{a}_m, q_{m-1}>$
0	0	$<\vec{a}_m, \vec{q}_m>$

The i, jth entry of R is $< \vec{a}_i , \vec{q}_j >$

where i , j = 1, … , m

Application

A main application of QR decomposition stems from the least squares problem. In these types of problems, you are given a non-linear system of equations, where the matrix in question may not necessarily be a square matrix (which is why the definition of QR decomposition we focus on does not necessarily have N = M).

The least squares problem is meant to solve $A\vec{x} = \vec{b}$, where A is the same N x M matrix with $\vec{x}$ as M x 1 and $\vec{b}$ is N x 1, which is meant to approximate a line of best fit. Since A = QR,

$$A\vec{x} = \vec{b}$$

$$(QR)\,\vec{x} = \vec{b}$$

$$R\vec{x} = Q^T \vec{b}$$

Note: This is derived from an estimator known as “Ordinary Least Squares” estimator, which involves the inverse of the matrix A.

Through least squares, we try to find an $\vec{x}$ that is close enough to solving the equation

$$A\vec{x} = \vec{b}$$

After plugging in Q & R into the above equation, we can then use *back-substitution* to get our solution $\vec{x}$

Homework Assignment

1) **Problem 1 (Multi-Part)**

Consider the 3x3 matrix A:

3	1	2
-2	0	3
0	-2	1

Calculate the QR Decomposition of A:

A) Use the Gram-Schmidt process to find Q:

a) What does $\vec{q}_1$ look like?

b) What does $\vec{q}_2$ look like?

c) What does $\vec{q}_3$ look like?

B) Find R

C) Given $A\vec{x} = \vec{y}$, where $\vec{y} = [5, 4, 0]^T$, find $\vec{x}$ using the given QR decomposition

2) **Problem 2 (Programming)**

Consider a 3x2 matrix B:

2	-2
4	3
1	-1

A) Write a program to calculate its QR decomposition (Hint: if writing in Python - *import scipy.linalg* and use the QR function)

a) Output your result

B) Write a program to verify your answers for problem 1a,b,c (i.e. write a method for back-substitution after you finding the QR decomposition)

a) Note: Don't forget to transpose Q

Homework Solutions

1. **Problem 1**

Remember, calculating each $\vec{q}$ can be determined through use of taking dot products and subtractions on repeat with the exception of $\vec{q_1}$:

a) What does $\vec{q_1}$ look like?

$$\vec{q_1} = \vec{a_1} / ||\vec{a_1}||_2$$

$$= [3\ -2\ 0]^T / \sqrt{3^2 + (-2)^2 + 0^2}$$

$$= [3\ -2\ 0]^T / \sqrt{13}$$

$$= [0.832050\ \ -0.55470019\ \ 0]^T$$

b) What does $\vec{q_2}$ look like?

$$\vec{q_2} = (\vec{a_2} - \frac{<\vec{a_2}, \vec{q_1}>}{<\vec{q_1}, \vec{q_1}>} \vec{q_1}) / ||(\vec{a_2} - \frac{<\vec{a_2}, \vec{q_1}>}{<\vec{q_1}, \vec{q_1}>} \vec{q_1})||_2$$

Note: $< \vec{q_1}, \vec{q_1} > = 1$

Numerator of $\vec{q_2}$:

$$([1\ \ 0\ \ -2\ ]^T - ([1\ \ 0\ \ -2\ ]\ [\frac{3}{\sqrt{13}}\ \ \frac{-2}{\sqrt{13}}\ \ 0]^T)*[\frac{3}{\sqrt{13}}\ \ \frac{-2}{\sqrt{13}}\ \ 0]^T)$$

$$= [1\ \ 0\ \ -2\ ]^T - (\frac{3}{\sqrt{13}})\ [\frac{3}{\sqrt{13}}\ \ \frac{-2}{\sqrt{13}}\ \ 0]^T$$

$$= [1 \;\; 0 \;\; -2]^T - [\frac{9}{13} \;\; \frac{-6}{13} \;\; 0]^T$$

$$= [\frac{4}{13} \;\; \frac{6}{13} \;\; -2]^T$$

Denominator of $\vec{q_2}$:

$$\sqrt{(\frac{4}{13})^2 + (\frac{6}{13})^2 + (-2)^2}$$

$$= 2.07549808$$

Thus,

$$\vec{q_2} = [\frac{4}{13} \;\; \frac{6}{13} \;\; -2]^T / 2.07549808$$

$$= [0.1482498 \;\; 0.2223747 \;\; -0.963624]^T$$

c) What does $\vec{q_3}$ look like?

$$\vec{q_3} = (\vec{a_3} - \frac{<\vec{a_3}, \vec{q_1}>}{<\vec{q_1}, \vec{q_1}>} \vec{q_1} - \frac{<\vec{a_3}, \vec{q_2}>}{<\vec{q_2}, \vec{q_2}>} \vec{q_2}) / ||\vec{a_3} - \frac{<\vec{a_3}, \vec{q_1}>}{<\vec{q_1}, \vec{q_1}>} \vec{q_1} - \frac{<\vec{a_3}, \vec{q_2}>}{<\vec{q_2}, \vec{q_2}>} \vec{q_2}||_2$$

Numerator of $\vec{q_3}$:

$$= [2 \;\; 3 \;\; 1]^T - ([2 \;\; 3 \;\; 1] [0.1482498 \;\; 0.2223747 \;\; -0.963624]^T)*[0.1482498$$

$$0.2223747 \;\; -0.963624]^T) - ([2 \;\; 3 \;\; 1] [\frac{3}{\sqrt{13}} \;\; \frac{-2}{\sqrt{13}} \;\; 0]^T)*[\frac{3}{\sqrt{13}} \;\; \frac{-2}{\sqrt{13}} \;\; 0]^T)$$

$$= [2 \;\; 3 \;\; 1]^T - (0.2964996+0.6671241 + -0.963624)*[0.1482498 \;\; 0.2223747 \;\; -$$

$$0.963624]^T) - (\frac{6}{\sqrt{13}} + \frac{-6}{\sqrt{13}} + 0])*[\frac{3}{\sqrt{13}} \;\; \frac{-2}{\sqrt{13}} \;\; 0]^T)$$

$$= [2 \;\; 3 \;\; 1]^T - [0 \;\; 0 \;\; 0]^T - [0 \;\; 0 \;\; 0]^T$$

$$= [2 \;\; 3 \;\; 1]^T$$

Denominator of $\vec{q_3}$:

$$\sqrt{(2)^2 + (3)^2 + (1)^2}$$

$$= \sqrt{4 + 9 + 1}$$

$$= \sqrt{14}$$

Thus,

$$\vec{q_3} = [2 \;\; 3 \;\; 1]^T / \sqrt{14}$$

$$= [0.5345224838 \;\; 0.8017837257 \;\; 0.2672612419 \;]^T$$

Q is:

0.832050	0.1482498	0.5345224838
-0.55470019	0.2223747	0.8017837257
0	-0.963624	0.2672612419

Now for R, recall R is the coefficients of each of the projections, namely:

$< \vec{a_1}, \vec{q_1} >$	$< \vec{a_2}, \vec{q_1} >$	$< \vec{a_3}, \vec{q_1} >$
0	$< \vec{a_2}, \vec{q_2} >$	$< \vec{a_3}, \vec{q_2} >$
0	0	$< \vec{a_3}, \vec{q_3} >$

Therefore from our earlier calculations, R is:

Row 1:

$$< \vec{a_1}, \vec{q_1} > = [3 \;\; -2 \;\; 0 \;] \cdot [0.832050 \;\; -0.55470019 \;\; 0]^T$$

$$= 3.60555038$$

$$< \vec{a_2}, \vec{q_1} > = [1 \;\; 0 \;\; -2] \cdot [0.832050 \; -0.55470019 \;\; 0]^T$$

$$= 0.832050$$

$$< \vec{a}_3 , \vec{q}_1 > = [2 \;\; 3 \;\; 1] \cdot [0.832050 \;\; -0.55470019 \;\; 0]^T$$

$$= 0$$

Row 2:

$$< \vec{a}_2 , \vec{q}_2 > = [1 \;\; 0 \;\; -2 \;] \cdot [0.1482498 \;\; 0.2223747 \;\; -0.963624 \;]^T$$

$$= 2.0754978$$

$$< \vec{a}_3 , \vec{q}_2 > = [2 \;\; 3 \;\; 1 \;] \cdot [0.1482498 \;\; 0.2223747 \;\; -0.963624 \;]^T$$

$$= 0$$

Row 3:

$$< \vec{a}_3 , \vec{q}_3 > = [2 \;\; 3 \;\; 1 \;] \cdot [0.5345224838 \;\; 0.8017837257 \;\; 0.2672612419 \;]^T$$

$$= 3.74165738$$

3.60555038	0.832050	0
0	2.0754978	0
0	0	3.74165738

For $A\vec{x} = \vec{y}$, where $\vec{y} = [5, 4, 0]^T$, finding $\vec{x}$ using the given QR decomposition utilizes the simplified least squares estimator derivation:

$$R\vec{x} = Q^T \vec{y}$$

Reminder: Q^T means every column of Q becomes a row of Q^T and every row of Q becomes a respective column of Q^T

$\vec{y} = [5, 4, 0]^T$

Q^T =

0.832050	-0.55470019	0
0.1482498	0.2223747	-0.963624
0.5345224838	0.8017837257	0.2672612419

$$R\vec{x} = Q^T \vec{y}$$

$$Q^T \vec{y} =$$

$$[5*(0.832050) + 4*(-0.55470019) + 0*0$$

$$5*(0.1482498) + 4*(0.2223747) + 0*(-0.963624)$$

$$5*(0.5345224838) + 4*(0.8017837257) + 0*(0.2672612419)]$$

$$= [1.94144924 \;\; 1.6307478 \;\; 5.8797473218]^T$$

$\vec{x}$ = $[x_1 \; x_2 \; x_3]^T$

R =

3.60555038	0.832050	0
0	2.0754978	0
0	0	3.74165738

Our system of equations formulated from above:

$3.60555038\, x_1 + 0.832050\, x_2 = 1.94144924$

$2.0754978\, x_2 = 1.6307478$

$3.74165738\, x_3 = 5.8797473218$

We can easily solve for $x_1 \; x_2 \; x_3$:

$x_2 = \frac{1.6307478}{2.0754978} = 0.7857140586$

$x_3 = \frac{5.8797473218}{3.74165738} = 1.5714285742$

$\Rightarrow 3.60555038\, x_1 + 0.832050*(0.7857140586) = 1.94144924$

$\Rightarrow 3.60555038\, x_1 + 0.6537533825 = 1.94144924$

$\Rightarrow 3.60555038\, x_1 = 1.2876958575$

$x_1 = \frac{1.2876958575}{3.60555038} = 0.3571426611$

Therefore our solution is: $\vec{x}$ = [0.3571426611 0.7857140586 1.5714285742]^T

2. Problem 2

The following code is written in Python and utilizes the library for calculating QR decomposition:

Due to the way the QR decomposition was implemented in Python, signs may differ.

```
import scipy
from scipy.linalg import qr
import numpy as np
import pprint

B = np.array([[2, -2], [4, 3], [1, -1]])
Q, R = qr(B)

print("B:")
pprint.pprint(B)

print ("Q:")
pprint.pprint(Q)

print ("R:")
pprint.pprint(R)

print("\n----------\n")
print("Problem 1 Verification")
A = np.array([[3,1,2], [-2,0,3], [0,-2,1]])
Q, R = qr(A)
```

```
print("A:")
pprint.pprint(A)

print ("Q:")
pprint.pprint(Q)

print ("R:")
pprint.pprint(R)

y = np.array([[5],[4],[0]])
print("\ny: \n",y)
Qty = np.matmul(Q.transpose(),y)
print("\nQ transpose * y:\n",Qty)

x_un = np.zeros(3)

#Back-Subtitution
n=3 #Dimension 3
for i in range(n-1, -1, -1):
    temp = Qty[i]
    for j in range(n-1, i, -1):
        temp = temp - x_un[j]*R[i,j]

    x_un[i] = temp/R[i,i]

print("\nSolution: ",x_un)
```

The back substitution code consists of two for loops, both decrementing back row by row, setting a temporarily variable to store each calculation until finally reassigning to the respective component of x_un, which is our unknown vector and will serve as the vector that is closest to the given y.

```
B:
array([[ 2, -2],
       [ 4,  3],
       [ 1, -1]])
Q:
array([[-4.36435780e-01,  7.80720058e-01, -4.47213595e-01],
       [-8.72871561e-01, -4.87950036e-01, -4.95102588e-18],
       [-2.18217890e-01,  3.90360029e-01,  8.94427191e-01]])
R:
array([[-4.58257569, -1.52752523],
       [ 0.        , -3.41565026],
       [ 0.        ,  0.        ]])



Problem 1 Verification
A:
array([[ 3,  1,  2],
       [-2,  0,  3],
       [ 0, -2,  1]])
Q:
array([[-0.83205029, -0.14824986,  0.53452248],
       [ 0.5547002 , -0.22237479,  0.80178373],
       [-0.        ,  0.96362411,  0.26726124]])
R:
array([[-3.60555128e+00, -8.32050294e-01,  0.00000000e+00],
       [ 0.00000000e+00, -2.07549809e+00,  8.88178420e-16],
       [ 0.00000000e+00,  0.00000000e+00,  3.74165739e+00]])

y:
 [[5]
 [4]
 [0]]

Q transpose * y:
 [[-1.94145069]
 [-1.6307485 ]
 [ 5.87974732]]

Solution:  [0.35714286 0.78571429 1.57142857]
```

Homework Narrative Reflection

The first problem demonstrates and walks a student through calculating each individual column of the newly created Q matrix is orthonormal as described in the definition of what it means to be part of the QR decomposition. For those unfamiliar with the Gram-Schmidt process, it can be a significant amount of information to digest and work through. The first homework problem helps apply the definition by breaking down Q and R into steps, especially considering there is significant calculation to be aware of for each of the columns to ensure they are orthonormal for Q.

The second problem demonstrates how QR decomposition can be better than LU decomposition because we consider not only square matrices but also rectangular ones as discussed in the introduction. The second problem also demonstrates the application of least squares to a non-linear system of equations. By applying to rectangular matrices, this reinforces that QR decomposition has more power than LU decomposition and that every matrix could have a QR decomposition.

Conclusion

QR Decomposition allows for more flexibility in decomposing given matrices, allowing for a matrix to be decomposed in a way where LU decomposition may fail. QR decomposition also allows for review of projections and Gram-Schmidt process, as well as what it means to be orthonormal, all of which are significant concepts in Linear Algebra. An interesting connection to this was that if Q is a square matrix, and since the columns are orthonormal, then this implies that Q is a unitary matrix, preserving the inner product and also implies that the conjugate

transpose of Q is equal to its inverse. This is why the computation for the least squares estimator using QR decomposition is easy to compute, especially considering that the matrix I gave was only consisting of real entries, then the conjugate transpose is simply the transpose.

This project was definitely a learning curve in terms of trying to connect linear algebra concepts more deeply to numerical analysis. What made the project fun for me was the fact that I was simultaneously taking MATH 103 (Advanced Linear Algebra) at the same time as this class, so getting to learn more in-depth about Gram-Schmidt, projections, as well as unitary matrices and seeing how they applied here made the whole experience both enjoyable and tedious.

Bibliography

1. Taboga, Marco (2021). "QR decomposition", Lectures on matrix algebra. https://www.statlect.com/matrix-algebra/QR-decomposition.
2. Shaffer, Ben (2020). “QR Matrix Factorization” https://towardsdatascience.com/qr-matrix-factorization-15bae43a6b2
3. Documentation on Python Library utilizing QR Decomposition: https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.qr.html