QR FACTORIZATION

David F. Gleich

August 21, 2023

1 LEAST SQUARES VIA QR FACTORIZATION AND ORTHOGONALIZATION

Learning objectives
1. Target pieces of a matrix for an operation with pieces of the identity matrix.

There is another approach to solving the least squares problems

$$\text{minimize} \quad \|\mathbf{b} - \boldsymbol{A}\mathbf{x}\|$$

besides the variable elimination procedure we saw in previous classes. I don't yet have a natural derivation of this particular idea, but I believe it originates around the following set of ideas.

- The geometry of the least squares problems involves working with the span of $\boldsymbol{A}$'s columns, or the range of $\boldsymbol{A}$. In particular, we want to find a point in the range that is as close as possible to $\mathbf{b}$.
- Since this involves working with the range of $\boldsymbol{A}$, it is "natural' ' to seek an orthogonal basis for it.

And this is what the QR factorization of a matrix encodes: an orthogonal basis for the columns of $\boldsymbol{A}$.

More formally, the QR factorization of a tall $m \times n$ matrix $\boldsymbol{A}$ (with $m \geq n$) is a pair of matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ such that:

- $\boldsymbol{A} = \boldsymbol{QR}$
- $\boldsymbol{Q}$ is square $m \times m$ and orthogonal
- $\boldsymbol{R}$ will also be upper-triangular and $m \times n$, but let's see where that comes from!

The upper-triangular structure appears to arise from early work by Schmidt on orthogonalizing a set of vectors. This is often called the "Gram-Schmidt process" and functions by successive orthogonalization.[1]

[1] I am looking into ways of re-deriving these ideas where the upper-triangular structure is one of a few possible natural choices depending on the ideas involved, but so far I haven't hit on anything easy.

1.1 REVIEW OF GRAM-SCHMIDT

This review is meant to remind you of stuff you hopefully learned in previous linear algebra classes.

That is, if we are given a set of three vectors $\mathbf{x}, \mathbf{y}, \mathbf{z}$ then the Gram-Schmidt process builds an orthonormal basis for their span, which is equivalent to building an orthogonal matrix $\boldsymbol{Q}$ such that

$$\begin{bmatrix} \mathbf{x} & \mathbf{y} & \mathbf{z} \end{bmatrix} = \boldsymbol{QC}$$

for some non-singular, square matrix $\boldsymbol{C}$. The Gram-Schmidt process begins with the first vector $\mathbf{x}$ and sets the first column of $\boldsymbol{Q}$ to be $\mathbf{x}/\|\mathbf{x}\|$. Then we *project-out* any component of $\mathbf{x}$ on the other vectors. The matrix $\boldsymbol{P}(\mathbf{x}) = \boldsymbol{I} - \frac{\mathbf{x}\mathbf{x}^T}{\mathbf{x}^T\mathbf{x}}$ is a projector[2] to the space orthogonal to the vector $\mathbf{x}$. That is, $\mathbf{x}^T\boldsymbol{P}(\mathbf{x})\mathbf{y} = \mathbf{x}^T\mathbf{y} - \frac{\mathbf{x}^T\mathbf{x}}{\mathbf{x}^T\mathbf{x}}\mathbf{x}^T\mathbf{y} = 0$. Hence, we compute $\mathbf{y}_1 = \boldsymbol{P}(\mathbf{x})\mathbf{y}$, $\mathbf{z}_1 = \boldsymbol{P}(\mathbf{x})\mathbf{z}$. The next vector $\mathbf{q}_2 = \mathbf{y}_1/\|\mathbf{y}_1\|$. and we project $\mathbf{z}_2$ via $\boldsymbol{P}(\mathbf{y}_1)$. This gives us three vectors:

$$\boldsymbol{Q} = \begin{bmatrix} \mathbf{x}/\|\mathbf{x}\| & \mathbf{y}_1/\|\mathbf{y}_1\| & \mathbf{z}_2/\|\mathbf{z}_2\| \end{bmatrix}$$

[2] A projector matrix *projects* vectors to a subspace S. Because the output from a projector is a new vector in a subspace S, it must be the case that projecting to S again will leave the result unchanged. Hence, $\boldsymbol{P}^2 = \boldsymbol{P}$ for any projector matrix!

where $\mathbf{y}_1 = \boldsymbol{P}(\mathbf{x})\mathbf{y}$ and $\mathbf{z}_2 = \boldsymbol{P}(\mathbf{y}_1)\boldsymbol{P}(\mathbf{x})\mathbf{z}$. We can write this as a matrix equation as follows:

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{x} & \mathbf{y} & \mathbf{z} \end{bmatrix} = \begin{bmatrix} \mathbf{x}/\|\mathbf{x}\| & \mathbf{y}_1/\|\mathbf{y}_1\| & \mathbf{z}_2/\|\mathbf{z}_2\| \end{bmatrix} \begin{bmatrix} \|\mathbf{x}\| & C_{1,2} & C_{1,3} \\ 0 & \|\mathbf{y}_1\| & C_{2,3} \\ 0 & 0 & \|\mathbf{z}_2\| \end{bmatrix}$$

where $C_{i,j}$ arises from the projection operations. Consider $C_{1,2}$, which we get from $\mathbf{y}_1 = \boldsymbol{P}(\mathbf{x})\mathbf{y} = \mathbf{y} - \frac{\mathbf{x}^T\mathbf{y}}{\mathbf{x}^T\mathbf{x}}\mathbf{x}$, we can write this to get $C_{i,j}$ for each.

Notice the similarity between this procedure and the successive elimination procedure we had in the previous class. I think this can be turned into a fairly natural derivation, but it requires a little more work.

The point of these derivations is that the Gram-Schmidt process produces an orthogonal basis for the columns of $\boldsymbol{A}$ via successive orthogonalization, which can be written:

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}$$

for an $m \times n$ matrix $\boldsymbol{Q}$ and a square upper-triangular matrix $n \times n$ matrix $\boldsymbol{R}$. This is often called a "thin' ' QR factorization because the matrix $\boldsymbol{Q}$ isn't square but is *tall* instead.

1.2 GENERALIZING TO QR

The idea with the full QR factorization is that we can extend a "thin" QR factorization to a square matrix $\boldsymbol{Q}$ because there are n orthogonal vectors in an n-dimensional space. Given any set of m orthogonal vector (say via Gram-Schmidt), then there exist another $m - n$ vectors that are mutually orthgonal as well. Of course, because these are orthogonal, we don't need to use them to write the matrix $\boldsymbol{A}$, so the "tail' ' of $\boldsymbol{R}$ becomes zero.

1.3 USING QR TO SOLVE LEAST SQUARES

Now, let's show that we can use *any* QR factorization to compute a solution to the least squares problem. Note that $\|\mathbf{x}\| = \|\boldsymbol{Q}\mathbf{x}\| = \|\boldsymbol{Q}^T\mathbf{x}\|$ for any square orthogonal matrix $\boldsymbol{Q}$.

Hence, let $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}$ be any full QR factorization with a square matrix $\boldsymbol{Q}$, then

$$\|\mathbf{b} - \boldsymbol{A}\mathbf{x}\| = \|\boldsymbol{Q}^T\mathbf{b} - \boldsymbol{Q}^T\boldsymbol{A}\mathbf{x}\| = \|\hat{\mathbf{b}} - \boldsymbol{R}\mathbf{x}\| = \left\| \begin{bmatrix} \hat{\mathbf{b}}_1 \\ \hat{\mathbf{b}}_2 \end{bmatrix} - \begin{bmatrix} \boldsymbol{R}_1 \\ 0 \end{bmatrix} \mathbf{x} \right\|.$$

Here, we used $\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_1 \\ 0 \end{bmatrix}$ where $\boldsymbol{R}_1$ is the first set of n rows of $\boldsymbol{R}$. Because $\boldsymbol{R}$ is upper-triangular, the other elements are always zero.

Note that this form helps us greatly! Note that no matter how we change $\mathbf{x}$, we cannot elminate $\hat{\mathbf{b}}_2$ from the difference between $\mathbf{b}$ and $\boldsymbol{A}\mathbf{x}$. Hence, the best we can do to minimize the expression is to set $\mathbf{x}$ so that $\hat{\mathbf{b}}_1 = \boldsymbol{R}_1\mathbf{x}$.

Consequently, we can use any method to produce a QR factorization to solve a least squares problem via the following algorithm:

```
Compute a full or thin QR factorization.
Compute b̂₁ = first n rows of Qb when Q is full,
  or b̂₁ = Q^T b when Q is m × n.
Solve R₁x = b̂₁.
Return x
```

1.4 A GIVENS ROTATIONS AND QR FOR A SMALL VECTOR.

Consider the problem of computing a QR factorization for a 2×1 vector $\mathbf{v}$. Recall that an orthogonal matrix is a generalization of a rotation, so we can write it as:

$$\boldsymbol{Q} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}.$$

Let's see how to pick $\boldsymbol{Q}$ for $\mathbf{v}$.

An obvious way is to try and compute θ in the above expression such that

$$\boldsymbol{Q}(\theta)\mathbf{v} = \gamma\mathbf{e}_1$$

for some γ.

However, there is a better way to do this! Note that $\boldsymbol{Q}(\theta)$ only has two unknowns, $c = \cos\theta$ and $s = \sin\theta$. To compute $\mathbf{Q}$, we just need these two values! Let's write out the equations:

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \gamma \\ 0 \end{bmatrix}$$

This gives two equations and two unknowns.

$$v_1 c + v_2 s = \gamma \text{ and } v_2 c - v_1 s = 0.$$

We can solve these to get[3] Some discussion of how this impacts numerical software is }

$$c = v_1/\gamma \text{ and } s = v_2/\gamma.$$

Because the matrix is orthgonal, we must have $\gamma = \sqrt{v_1^2 + v_2^2}$ or $\gamma = -\sqrt{v_1^2 + v_2^2}$ so that the length of $\mathbf{v}$ doesn't change.

This 2×2 matrix $\boldsymbol{Q}(\theta)$ is called a Givens rotation.

[3] The solution here is not unique. Note that we can negate these values as well as they are also a solution. See more discussion in `https://netlib.org/lapack/lawnspdf/lawn148.pdf`

1.5 THE QR FACTORIZATION FOR A 3X1 VECTOR.

Suppose $\mathbf{v}$ is 3×1. Then we *could* seek to build a 2d rotation matrix and solve for the coefficients. However, there is an alternative mechanism where we can use matrix structure. Let $\mathbf{v} = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}^T$. Let $

1.6 GIVENS ROTATIONS IN JULIA

We can compute Givens rotations in J

1.7 COMPUTING QR FOR A COLUMN

Consider computing a QR factorization for a $n \times 1$ vector $\mathbf{v}$ now. By the definition, we have:

$$\mathbf{Q}\mathbf{v} = \gamma \mathbf{e}_1.$$

where $\gamma = \pm\|\mathbf{v}\|$.