

The QR Factorization

Gaussian elimination, and the closely-related LU and PLU factorizations, provide one approach to solving a linear system $Ax = b$. This approach is based on using elementary matrices, which simply implement row operations. A much newer approach is the *QR factorization*, which is based on *Householder* matrices. We are interested in this algorithm because it will provide some nice examples of algorithm development and analysis.

The Householder Matrix

The following develops a special utility matrix that, like the row operation-based matrices, can be used to perform a matrix factorization.

Consider a matrix H of the form

$$H = I - \sigma uu^T,$$

where I is the n by n identity matrix, u is an n by 1 matrix—basically a vector—and σ is some fixed but unknown real number.

- △ Compute by hand the matrix H that results from $u = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ and $\sigma = 1$.

Now we want to figure out how to pick u and σ to make H especially nice.

First, note that H is symmetric, no matter what values u and σ have. To see this, just use basic properties of the transpose operation, namely $(A^T)^T = A$, $(A + B)^T = A^T + B^T$, $(AB)^T = B^T A^T$, and $(\alpha A)^T = \alpha A^T$ where A and B are any matrices for which the given operations are defined, and α is any real number.

Specifically,

$$H^T = (I - \sigma uu^T)^T = I^T - (\sigma uu^T)^T = I - \sigma (uu^T)^T = I - \sigma (u^T)^T u^T = I - \sigma uu^T = H.$$

- △ Are the elementary matrices used in the PLU factorization symmetric? Are the inverses of the elementary matrices easy to compute?

Next, we want to figure out how to pick σ to make H have a nice inverse. It turns out that we can pick σ so that H is its own inverse, namely $HH = I$.

The following derivation uses basic matrix algebra properties, which say that you can do algebra just like you're used to, except that you must keep things that are multiplied together in the correct order. And, you can commute *scalars* with matrices freely.

Leaving out a lot of basic algebra steps, we have

$$HH = (I - \sigma uu^T)(I - \sigma uu^T) = I + (\sigma^2 u^T u - 2\sigma)uu^T,$$

so if we pick σ to make

$$\sigma^2 u^T u - 2\sigma = 0,$$

we have $\sigma = 0$ or $\sigma = \frac{2}{u^T u}$. Clearly $\sigma = 0$ does yield a nice H , namely $H = I$, but that's not very helpful, so we take $\sigma = \frac{2}{u^T u}$.

We now officially define a Householder matrix for a given vector u to be

$$H = I - \frac{2}{u^T u} u u^T.$$

△ Let $u = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ and compute the corresponding Householder matrix. Verify directly that $H^T = H$ and $HH = I$.

We're almost done with the preliminaries before we can define the QR factorization. Now we reach the crucial point, namely picking u so that the corresponding Householder matrix “zeroes out” a given column of a matrix below the diagonal.

Specifically, let v (for “victim,” perhaps) be any vector and assume that we want to find

a Householder matrix H such that $Hv = \begin{bmatrix} \beta \\ 0 \\ \vdots \\ 0 \end{bmatrix}$.

Let's just form Hv and see if we can figure out how to do this:

$$Hv = (I - \sigma u u^T)v = v - \sigma u^T v u.$$

△ This innocent-looking calculation also shows how Hv can be efficiently computed for any vector v . In general, what is the efficiency class for multiplying an n by n matrix by an n vector? What is the efficiency class for the algorithm suggested by the preceding calculation?

Now, looking at the formula above, with our favorite value for σ plugged in, we have

$$Hv = v - \frac{2u^T v}{u^T u} u.$$

If we want that quantity to have all zeroes, except in the first spot, one obvious thing suggests itself: take u to match v in positions 2 through n , and fiddle with u_1 to make the scalar equal to 1.

So, we want to solve the equation

$$\frac{2u^T v}{u^T u} = 1$$

for α , where

$$u = \begin{bmatrix} \alpha \\ v_2 \\ \vdots \\ v_n \end{bmatrix}.$$

After some algebraic suffering, we obtain the equivalent equation

$$\alpha^2 - 2v_1\alpha - v_2^2 - \dots - v_n^2 = 0,$$

which is a quadratic equation with unknown α and has the solutions

$$\alpha = \frac{2v_1 \pm \sqrt{4v_1^2 + 4(v_2^2 + \dots + v_n^2)}}{2}.$$

This simplifies in convincing fashion to

$$\alpha = v_1 \pm \sqrt{v_1^2 + v_2^2 + \dots + v_n^2}.$$

Either solution does what we want in general, so we can pick the sign to match the sign of v_1 .

So, the upshot of this whole discussion is that if we have a matrix A that we want to transform, we begin by forming a vector u that is identical to the first column of A except in the first position, where it has the original value of A with the sum of the squares of all the components of the first column added or subtracted.

Then we take that u , compute the corresponding σ , and the resulting H is the matrix we multiply A by zero it out below the diagonal, just like in Gaussian elimination.

- △ Here's a major point of this discussion: how efficiently can HA be formed? Should H be formed? Should H be stored?

Then, the process is repeated, just like in Gaussian elimination, at step k zeroing out column k of $H_{k-1} \cdots H_2 H_1 A$ below row k , while leaving rows 1 through $k-1$ unchanged.

- △ Verify that forming u for column k by putting 0 in spots 1 through $k-1$ works as desired.

So, eventually we obtain

$$H_n \cdots H_2 H_1 A = R,$$

where R is upper triangular. Then we multiply both sides repeatedly by $H_n, H_{n-1}, \dots, H_2, H_1$, to obtain

$$A = H_1 H_2 \cdots H_n R$$

and denote $H_1 H_2 \cdots H_n$ by Q , so $A = QR$ as desired, where R is upper triangular and Q is, well, what about Q ?

It turns out that Q , as a product of Householder matrices, is a very nice matrix, having much better properties in general than the matrix L from Gaussian elimination.

Q has the very nice property that $Q^T Q = I$.

- △ Verify this.

This means that the columns of Q form a set of mutually perpendicular vectors of length 1 (in the usual Euclidean way of measuring the length of a vector). In fact, the QR factorization is a better method for finding an orthogonal basis for a subspace spanned by a given set of vectors than the much older and more famous “Gram Schmidt orthogonalization algorithm.”

Now, to solve $Ax = b$ once we have $A = QR$, we note that we want $Ax = QRx = b$, so if we multiply both sides by Q^T , we have $Q^TQRx = Q^Tb$, or $Rx = Q^Tb$. Thus, given a new right-hand side vector b , we first form Q^Tb , and then use back substitution to find x , which we can do because R is upper triangular.

At this point you might be expecting a nice example worked out, showing this process on, say, a 4 by 4 matrix, but you’re not going to get it, because there wouldn’t be much point! The deep fact here is that this algorithm always leads to horrible numbers, and only the tiniest of examples can be “rigged” to work out decently. This whole thing is really an advertisement for your education: by studying complex algorithms and mathematics in depth, you gain the ability to deal with algorithms that are simply too hideous to do by hand. You can’t understand the QR algorithm by just working through an example (though working through a small one would help a little), you can only understand it through abstract analysis, and by programming it up.

△ Having said all that, form by hand the QR factorization of the matrix

$$A = \begin{bmatrix} 2 & -2 & 1 \\ 1 & -24/5 & 2 \\ 2 & 7/5 & 3 \end{bmatrix}.$$

While doing this, think about how the QR factorization process can be efficiently implemented, and what its efficiency class will be.

So, it turns out that the QR approach is competitive, in terms of run-time efficiency, with Gaussian elimination with partial pivoting. But, it is typically better than that algorithm in the presence of ill-conditioning.

Also, it can be used as a tool for other purposes. As mentioned above, it can be used as a practical alternative to the Gram-Schmidt algorithm. It provides a typically superior alternative to the “normal equations” when doing least-squares regression. And, it is the core computational element in algorithms to compute eigenvalues.