QR Decomposition and Least Squares

We have discussed the *Gram-Schmidt algorithm* for constructing an orthonormal set of vectors from a linearly independent set. When we perform the Gram-Schmidt algorithm, we are actually factoring a matrix A in the form $A = QR$, where Q is a matrix with orthonormal columns and R is an upper triangular matrix. This factorization can be done in MATLAB by typing `[Q,R]=qr(A,0)`.

Example:

$$A = \begin{pmatrix} 1 & 1 & 2 \\ 1 & -1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Factor A in the form $A = QR$, where Q is an *orthogonal matrix* (i.e., a square matrix whose columns are *orthonormal*) and R is upper triangular.

The Gram-Schmidt algorithm for orthonormalizing the columns of A gives us such a factorization. The first step is to divide column one of A by its norm, to obtain the first orthonormal vector. This corresponds to multiplying A on the right by a diagonal matrix, which we will call R_1:

$$AR_1 = \begin{pmatrix} 1 & 1 & 2 \\ 1 & -1 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1/\sqrt{3} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{3} & 1 & 2 \\ 1/\sqrt{3} & -1 & 1 \\ 1/\sqrt{3} & 1 & 0 \end{pmatrix}.$$

The first column of this new matrix is our first orthonormal vector.

Next we replace column two of the matrix by itself minus a constant times column one, where the constant is chosen so that the new vector will be orthogonal to column one:

$$\begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} \leftarrow \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} - \frac{1}{\sqrt{3}} \begin{pmatrix} 1/\sqrt{3} \\ 1/\sqrt{3} \\ 1/\sqrt{3} \end{pmatrix}.$$

The constant $1/\sqrt{3}$ was determined by taking the inner product of column one with column two:

$$(1/\sqrt{3}, 1/\sqrt{3}, 1/\sqrt{3}) \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{3}}.$$

This corresponds to multiplying on the right by an upper triangular matrix R_2:

$$AR_1R_2 = \begin{pmatrix} 1/\sqrt{3} & 1 & 2 \\ 1/\sqrt{3} & -1 & 1 \\ 1/\sqrt{3} & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & -1/\sqrt{3} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{3} & 2/3 & 2 \\ 1/\sqrt{3} & -4/3 & 1 \\ 1/\sqrt{3} & 2/3 & 0 \end{pmatrix}.$$

Notice that the second column of the new matrix is orthogonal to the first.

Next we normalize the second column of the matrix. This again corresponds to multiplying on the right by a diagonal matrix R_3:

$$AR_1R_2R_3 = \begin{pmatrix} 1/\sqrt{3} & 2/3 & 2 \\ 1/\sqrt{3} & -4/3 & 1 \\ 1/\sqrt{3} & 2/3 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \sqrt{3}/(2\sqrt{2}) & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{3} & 1/\sqrt{6} & 2 \\ 1/\sqrt{3} & -2/\sqrt{6} & 1 \\ 1/\sqrt{3} & 1/\sqrt{6} & 0 \end{pmatrix}.$$

Note that column two is still orthogonal to column one, but now it has norm one. This is the second orthonormal vector produced by the Gram-Schmidt algorithm.

Column three still has not been changed. The next step is to orthogonalize it against columns one and two:

$$\begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix} \leftarrow \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix} - \sqrt{3} \begin{pmatrix} 1/\sqrt{3} \\ 1/\sqrt{3} \\ 1/\sqrt{3} \end{pmatrix} - 0 \begin{pmatrix} 1/\sqrt{6} \\ -2/\sqrt{6} \\ 1/\sqrt{6} \end{pmatrix}.$$

The coefficients $\sqrt{3}$ and 0 are obtained by taking the inner product of columns one and two, respectively, with column three:

$$(1/\sqrt{3}, 1/\sqrt{3}, 1/\sqrt{3}) \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix} = \sqrt{3} \quad (1/\sqrt{6}, -2/\sqrt{6}, 1/\sqrt{6}) \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix} = 0.$$

This corresponds to multiplying on the right by another upper triangular matrix R_4:

$$AR_1R_2R_3R_4 = \begin{pmatrix} 1/\sqrt{3} & 1/\sqrt{6} & 2 \\ 1/\sqrt{3} & -2/\sqrt{6} & 1 \\ 1/\sqrt{3} & 1/\sqrt{6} & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & -\sqrt{3} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{3} & 1/\sqrt{6} & 1 \\ 1/\sqrt{3} & -2/\sqrt{6} & 0 \\ 1/\sqrt{3} & 1/\sqrt{6} & -1 \end{pmatrix}.$$

Note that column three is now orthogonal to columns one and two.

The last step is to normalize column three. This corresponds to multiplying on the right by a diagonal matrix R_5:

$$AR_1R_2R_3R_4R_5 = \begin{pmatrix} 1/\sqrt{3} & 1/\sqrt{6} & 1 \\ 1/\sqrt{3} & -2/\sqrt{6} & 0 \\ 1/\sqrt{3} & 1/\sqrt{6} & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1/\sqrt{2} \end{pmatrix} = \begin{pmatrix} 1/\sqrt{3} & 1/\sqrt{6} & 1/\sqrt{2} \\ 1/\sqrt{3} & -2/\sqrt{6} & 0 \\ 1/\sqrt{3} & 1/\sqrt{6} & -1/\sqrt{2} \end{pmatrix}.$$

The columns of our final matrix are orthonormal. Thus we can write

$$AR_1R_2R_3R_4R_5 = Q,$$

where Q is an orthogonal matrix. If we define

$$R = (R_1R_2R_3R_4R_5)^{-1} = R_5^{-1}R_4^{-1} \cdots R_1^{-1}, \tag{1}$$

then R is upper triangular, since the inverse of each upper triangular matrix $R_1, \ldots, R_5$ is upper triangular and the product of upper triangular matrices is upper triangular. We thus have the factorization $A = QR$.

The matrix R in (1) is easy to compute, since the matrices $R_1, \ldots, R_5$ are either diagonal or else have ones on the main diagonal and nonzeros above the diagonal in just one column. To invert a diagonal matrix, just take the inverse of each diagonal element. To invert a matrix with ones on the main diagonal and nonzeros in just one column, take the negative of the entries in that column. We can also compute R by noting that since $A = QR$ and $Q^TQ = I$, we have $Q^TA = Q^TQR = R$. For our example,

$$R = Q^TA = \begin{pmatrix} \sqrt{3} & 1/\sqrt{3} & \sqrt{3} \\ 0 & 2\sqrt{2}/3 & 0 \\ 0 & 0 & \sqrt{2} \end{pmatrix}.$$

Least Squares Solution of Overdetermined Linear Systems

Suppose we have an *overdetermined* system of linear equations, that is, more equations than unknowns, so that the system cannot be satisfied exactly. An example is

$$\begin{aligned} x_1 + x_2 &= 2 \\ x_1 - x_2 &= 0 \\ 2x_1 + x_2 &= 4 \end{aligned}$$

If the overdetermined system is denoted $Ax \approx b$, where A is an m by n matrix with $m > n$, and b is a given m-vector, then we would like to find an approximate solution x^* that minimizes the residual norm $\|b - Ax\|$; that is

$$\|b - Ax^*\| \leq \|b - Ax\|, \quad \text{for all } x \in R^n.$$

This is called a *least squares* solution. In our example, we look for x_1^* and x_2^* to minimize

$$\left\| \begin{pmatrix} 2 \\ 0 \\ 4 \end{pmatrix} - \begin{pmatrix} 1 & 1 \\ 1 & -1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \right\| = \sqrt{(2 - x_1 - x_2)^2 + (0 - x_1 + x_2)^2 + (4 - 2x_1 - x_2)^2}.$$

If b lies in the *range* of A, then the equation $Ax = b$ has an exact solution; that is, the least squares solution x^* satisfies $\|b - Ax^*\| = 0$. If b lies outside the range of A, then the equation $Ax = b$ cannot be solved exactly. Instead, we find the closest vector b^* to b in the range of A and solve $Ax^* = b^*$. This is described further in the text. The vector b^* is characterized by the fact that $b - b^* = b - Ax^*$ is orthogonal to the range of A. This enables us to determine x^* by solving the *normal equations*: $A^T(b - Ax^*) = 0$ or

$$A^TAx^* = A^Tb. \tag{2}$$

For our example, the normal equations are

$$\begin{pmatrix} 6 & 2 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 10 \\ 6 \end{pmatrix},$$

yielding $x_1 = 9/7$, $x_2 = 8/7$.

We can also use the (reduced) QR-decomposition of A to solve the least squares problem. [There are actually two types of QR-decompositions for an m by n matrix. In the *full* QR-decomposition, Q is m by m and R is m by n. We will discuss only the *reduced* QR-decomposition, in which Q is an m by n matrix with orthonormal columns and R is n by n. In the MATLAB expression `qr(A,0)`, the second argument 0 indicates that it is the reduced QR-decomposition that we seek.] The columns of Q form an *orthonormal basis* for the range of A. (Do you see why?) Hence $b - Ax^*$ will be orthogonal to the range of A if and only if $Q^T(b - Ax^*) = 0$. Since $A = QR$ and $Q^TQ = I$, this equation becomes $Q^Tb - Q^T(QR)x^* = Q^Tb - (Q^TQ)Rx^* = 0$ or,

$$Rx^* = Q^Tb. \tag{3}$$

This equation is easy to solve since R is upper triangular. We can use the routine `back_solve` that was described in a previous set of notes.

For our example problem, the (reduced) QR-decomposition is

$$\begin{pmatrix} 1 & 1 \\ 1 & -1 \\ 2 & 1 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{6} & 2/\sqrt{21} \\ 1/\sqrt{6} & -4/\sqrt{21} \\ 1/\sqrt{6} & 1/\sqrt{21} \end{pmatrix} \begin{pmatrix} \sqrt{6} & 2/\sqrt{6} \\ 0 & \sqrt{7/3} \end{pmatrix},$$

and so the linear system (3) becomes

$$\begin{pmatrix} \sqrt{6} & 2/\sqrt{6} \\ 0 & \sqrt{7/3} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 10/\sqrt{6} \\ 8/\sqrt{21} \end{pmatrix}.$$

(You should check this.) Solving for x_1 and x_2, we again obtain $x_1 = 9/7$, $x_2 = 8/7$.

When solving a small least squares problem by hand, it is usually easier to solve the normal equations (2) than it is to compute the QR-decomposition. When working on a computer, however, least squares problems are usually solved by computing the QR-decomposition of the matrix and then backsolving to solve the n by n upper triangular system in (3). This is because the QR-decomposition is less susceptible to the effects of rounding errors and hence delivers a more accurate approximation to the least squares solution than does solving the normal equations. This is what MATLAB does when you enter a nonsquare matrix A (m by n, say, with $m \neq n$) and issue the command `A\b`. Recall that when A is square, MATLAB attempts to solve the linear system by Gaussian elimination and issues a warning if the matrix is (nearly) singular. In the nonsquare case, however, it uses the QR-decomposition and returns a least squares solution, even if the linear system cannot be solved exactly.