Notes for 2016-09-30

1 Logistics

1. HW 3 has minor edits to problem 3.
2. Per your request, we will move future homeworks (and the midterm) to have Monday due dates.

2 QR and Gram-Schmidt

We now turn to our first numerical method for computing the QR decomposition: the Gram-Schmidt algorithm. This method is usually presented in first linear algebra classes, but is rarely interpreted as a matrix factorization. Rather, it is presented as a way of converting a basis for a space into an orthonormal basis for the same space. If $a_1, a_2, \ldots, a_n$ are column vectors, the Gram-Schmidt algorithm is as follows: for each $j = 1, \ldots, n$

$$\begin{aligned} \tilde{a}_j &= a_j - \sum_{i=1}^{j-1} q_i q_i^T a_j \\ q_j &= \tilde{a}_j / \|\tilde{a}\|_j. \end{aligned}$$

At the end of the iteration, we have that the q_j vectors are all mutually orthonormal and

$$\operatorname{span}\{a_1, \ldots, a_j\} = \operatorname{span}\{q_1, \ldots, q_j\}.$$

To see this as a matrix factorization, we rewrite the iteration as

$$\begin{aligned} r_{ij} &= q_i^T a_j \\ \tilde{a}_j &= a_j - \sum_{i=1}^{j-1} q_i r_{ij} \\ r_{jj} &= \|\tilde{a}\|_j \\ q_j &= \tilde{a}_j / r_{jj} \end{aligned}$$

Putting these equations together, we have that

$$a_j = \sum_{i=1}^{j} q_i r_{ij},$$

or, in matrix form,

$$A = QR$$

where A and Q are the matrices with column vectors a_j and q_j, respectively.

Sadly, the Gram-Schmidt algorithm is not backward stable. The problem occurs when a vector a_j is nearly in the span of previous vectors, so that cancellation rears its ugly head in the formation of $\tilde{a}_j$. The classical Gram-Schmidt (CGS) method that we have shown is particularly problematic; a somewhat better alternative is the modified Gram-Schmidt method (MGS) algorithm:

```
% Overwrite A with Q via MGS, store R separately
R = zeros(n);
for j = 1:n
  for i = 1:n-1
    R(i,j) = Q(:,i)'*A(i,j);
    A(:,j) = A(:,j) - Q(:,i)*R(i,j);
  end
  R(j,j) = norm(A(:,j));
  A(:,j) = A(:,j) / R(j,j);
end
```

Though equivalent in exact arithmetic, the MGS algorithm has the advantage that it computes dot products with the updated $\tilde{a}_j$ as we go along, and these intermediate vectors have smaller norm than the original vector. Sadly, this does not completely fix the matter: the computed q_j vectors can still drift away from being orthogonal to each other. One can explicitly re-orthogonalize vectors that drift away from orthogonality, and this helps further. In practice, though, we usually don't bother: if backward stability is required, we turn to other algorithms.

Despite its backward instability, the Gram-Schmidt algorithm forms a very useful building block for iterative methods, and we will see it frequently in later parts of the course.

3 Householder transformations

The Gram-Schmidt orthogonalization procedure is not generally recommended for numerical use. Suppose we write $A = [a_1 \dots a_m]$ and $Q = [q_1 \dots q_m]$. The essential problem is that if $r_{jj} \ll \|a_j\|_2$, then cancellation can destroy the accuracy of the computed q_j; and in particular, the computed q_j may not be particularly orthogonal to the previous q_j. Actually, loss of orthogonality can build up even if the diagonal elements of R are not exceptionally small. This is Not Good, and while we have some tricks to mitigate the problem, we need a different approach if we want the problem to go away.

Recall that one way of expressing the Gaussian elimination algorithm is in terms of Gauss transformations that serve to introduce zeros into the lower triangle of a matrix. *Householder* transformations are orthogonal transformations (reflections) that can be used to similar effect. Reflection across the plane orthogonal to a unit normal vector v can be expressed in matrix form as

$$H = I - 2vv^T.$$

Now suppose we are given a vector x and we want to find a reflection that transforms x into a direction parallel to some unit vector y. The right reflection is through a hyperplane that bisects the angle between x and y (see Figure 1), which we can construct by taking the hyperplane normal to $x - \|x\|y$. That is, letting $u = x - \|x\|y$ and $v = u/\|u\|$, we have

$$\begin{aligned}(I - 2vv^T)x &= x - 2\frac{(x + \|x\|y)(x^Tx + \|x\|x^Ty)}{\|x\|^2 + 2x^Ty\|x\| + \|x\|^2\|y\|^2} \\ &= x - (x - \|x\|y) \\ &= \|x\|y.\end{aligned}$$

If we use $y = \pm e_1$, we can get a reflection that zeros out all but the first element of the vector x. So with appropriate choices of reflections, we can take a matrix A and zero out all of the subdiagonal elements of the first column.

Now think about applying a sequence of Householder transformations to introduce subdiagonal zeros into A, just as we used a sequence of Gauss transformations to introduce subdiagonal zeros in Gaussian elimination. This leads us to the following algorithm to compute the QR decomposition:

```
1 function [Q,R] = hqr1(A)
```

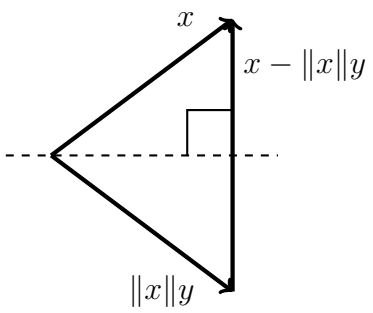

Figure 1: Construction of a reflector to transform x into $\|x\|y$, $\|y\| = 1$.

```
% Compute the QR decomposition of an m-by-n matrix A using
% Householder transformations.

[m,n] = size(A);
Q = eye(m);  % Orthogonal transform so far
R = A;       % Transformed matrix so far

for j = 1:n

  % -- Find H = I-tau*w*w' to put zeros below R(j,j)
  normx = norm(R(j:end,j));
  s    = -sign(R(j,j));
  u1   = R(j,j) - s*normx;
  w    = R(j:end,j)/u1;
  w(1) = 1;
  tau  = -s*u1/normx;

  % -- R := HR, Q := QH
  R(j:end,:) = R(j:end,:)-(tau*w)*(w'*R(j:end,:));
  Q(:,j:end) = Q(:,j:end)-(Q(:,j:end)*w)*(tau*w)';

end
```

Note that there are two valid choices of u_1 at each step; we make the choice that avoids cancellation in the obvious version of the formula.

As with LU factorization, we can re-use the storage of A by recognizing that the number of nontrivial parameters in the vector w at each step is the same as the number of zeros produced by that transformation. This gives us the following:

```
function [A,tau] = hqr2(A)
  % Compute the QR decomposition of an m-by-n matrix A using
  % Householder transformations, re-using the storage of A
  % for the Q and R factors.

  [m,n] = size(A);
  tau = zeros(n,1);

  for j = 1:n

    % -- Find H = I-tau*w*w' to put zeros below A(j,j)
    normx     = norm(A(j:end,j));
    s         = -sign(A(j,j));
    u1        = A(j,j) - s*normx;
    w         = A(j:end,j)/u1;
    w(1)      = 1;
    A(j+1:end,j) = w(2:end); % Save trailing part of w
    A(j,j)    = s*normx;    % Diagonal element of R
    tau(j)    = -s*u1/normx;

    % -- R := HR
    A(j:end,j+1:end) = A(j:end,j+1:end)-...
                  (tau(j)*w)*(w'*A(j:end,j+1:end));

  end
```

If we ever need Q or Q^T explicitly, we can always form it from the compressed representation. We can also multiply by Q and Q^T implicitly:

```
function QX = applyQ(QR,tau,X)

  [m,n] = size(QR);
  QX = X;
  for j = n:-1:1
    w = [1; QR(j+1:end,j)];
    QX(j:end,:) = QX(j:end,:)-(tau(j)*w)*(w'*QX(j:end,:));
  end
```

```
function QTX = applyQT(QR,tau,X)

  [m,n] = size(QR);
  QTX = X;
  for j = 1:n
    w = [1; QR(j+1:end,j)];
    QTX(j:end,:) = QTX(j:end,:)-(tau(j)*w)*(w'*QTX(j:end,:));
  end
```

4 Givens rotations

Householder reflections are one of the standard orthogonal transformations used in numerical linear algebra. The other standard orthogonal transformation is a *Givens rotation*:

$$G = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}.$$

where $c^2 + s^2 = 1$. Note that

$$G = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} cx - sy \\ sx + cy \end{bmatrix}$$

so if we choose

$$s = \frac{-y}{\sqrt{x^2 + y^2}}, \qquad c = \frac{x}{\sqrt{x^2 + y^2}}$$

then the Givens rotation introduces a zero in the second column. More generally, we can transform a vector in $\mathbb{R}^m$ into a vector parallel to e_1 by a sequence of $m - 1$ Givens rotations, where the first rotation moves the last element to zero, the second rotation moves the second-to-last element to zero, and so forth.

For some applications, introducing zeros one by one is very attractive. In some places, you may see this phrased as a contrast between algorithms based on Householder reflections and those based on Givens rotations, but this is not quite right. Small Householder reflections can be used to introduce one zero at a time, too. Still, in the general usage, Givens rotations seem to be the more popular choice for this sort of local introduction of zeros.