Week 7: Monday, Oct 5

Logistics

1. On HW2 p1: I swapped A and $\hat{A}$ from how they are in the notes. Arguably the notes are the more natural notation, so you can exchange $\hat{A}$ for A everywhere in the problem statement, if you wish.

2. On HW2 p3: You can compute $\|B^{-1}\|_\infty$ *exactly* without resorting to anything like Hager's estimator. Look at the relationship between rows of B^{-1} (you can derive a recurrence), and use that relationship in order to define a fast recurrence for the absolute row sums of B^{-1}.

3. On HW2 p4: The first part of the question is not whether $A + D$ is positive definite; it is why $LL^T = A+D$ where L and D are the matrices computed by the algorithm. For the second part of the question, you won't get particularly far if you write $D = UV^T$ where U and V are full size. The point of Sherman-Morrison-Woodbury is really to deal with low-rank updates written in outer product form, so that U and V are tall, skinny matrices.

   Also, note that MATLAB recognizes when a matrix is triangular. So if `L` is a triangular matrix, then

   ```
   x = L\b;
   ```

   is equivalent to computing $x = L^{-1}b$, and the computation runs in $O(n^2)$ time rather than the usual $O(n^3)$ time.

Linear least squares

Suppose $A \in \mathbb{R}^{m\times n}$ where $m > n$. Then in general we will not be able to solve systems of the form $Ax = b$, and the best we can do is to minimize the residual error. Minimizing in the 2-norm gives us the standard least squares problem:

$$\text{argmin}_x \|Ax - b\|_2^2.$$

Thing of the squared residual as a quadratic function in x:

$$F(x) = \|Ax - b\|^2 = (Ax - b)^T(Ax - b) = x^T A^T Ax - 2x^T A^T b + b^T b.$$

Then the minimum occurs when

$$\nabla F(x) = 2(A^T Ax - A^T b) = 0.$$

Thus we have

$$A^T Ax = A^T b.$$

These are the *normal equations*, so named because they are exactly the equations that make the residual $Ax - b$ orthogonal (normal) to anything vector Ay in the range space of A.

If A is full rank, then $A^T A$ is symmetric and positive definite matrix, and the normal equations have a unique solution that we can compute via Cholesky factorization. But $\kappa(A^T A) = \kappa(A)$, so if $\kappa(A)$ is even moderately large, the condition number for the normal equations may be terrible. We will therefore t

If A is rank-deficient, we have a *rank-deficient least squares* problem. Though (nearly) rank-deficient least squares problems are fairly common in practice, for the moment we will concentrate on the case when A has full rank.

Orthogonal transformations and Gram-Schmidt

Recall that orthogonal transformations have the property (and indeed can be defined by the property) that they leave the Euclidean norm alone. If Q is any matrix with orthonormal columns, we can write

$$\|Ax - b\|_2^2 = \|Q^T(Ax - b)\|_2^2 = \|Q^T Ax - Q^T b\|_2^2.$$

This suggests an alternative approach to the least squares problem: find Q such that $Q^T A$ has a relatively simple form. A natural choice is the decomposition

$$A = QR,$$

where Q is an $m \times m$ orthogonal matrix and R is an $m \times n$ upper triangular matrix. Eqvuivalently, we can write the "economy" version of the decomposition, $A = QR$ with an $m \times n$ matrix Q and an $n \times n$ upper triangular R, where the columns of Q form an orthonormal basis for the range space of A. Using this decomposition, we can solve the least squares problem via the triangular system

$$Rx = Q^T b.$$

The *Gram-Schmidt* procedure is usually the first method people learn for converting some existing basis (columns of A) into an orthonormal basis (columns of Q). For each column of A, the procedure subtracts off any components in the direction of the previous columns, and then scales the remainder to be unit length. In MATLAB, Gram-Schmidt looks something like this:

```
Q = [];
for j = 1:n
  v = A(:,j);      % Take the jth original basis vector
  v = v-Q*(Q'*v);  % Make it orthogonal to q_i, i = 1:j-1
  v = v/norm(v);   % Normalize what remains
  Q = [Q, v];      % Append the result to the basis
end
```

Where does R appear in this algorithm? It appears thus:

```
Q = [];
R = zeros(m);
for j = 1:n
  v = A(:,j);      % Take the jth original basis vector
  rp = Q'*v;       % Project v onto previous basis vectors
  v   = v-Q*u;     % Make vector orthogonal to q_i, i = 1:j-1
  rjj = norm(v);   % Get the normalizing factor
  v   = v/rjj;     % Normalize what remains
  Q   = [Q, v];    % Append the result to the basis
  R(1:j,j) = [rp; rjj]; % ... and update R
end
```

That is, R accumulates the multipliers that we computed from the Gram-Schmidt procedure. This idea that the multipliers in an algorithm can be thought of as entries in a matrix should be familiar, since we encountered it before when we looked at Gaussian elimination.

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

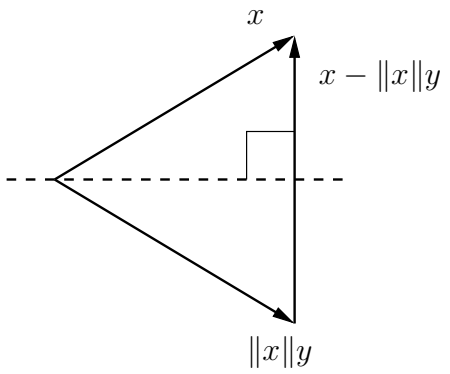

Figure 1: Construction of a reflector to transform x into $\|x\|y$, $\|y\| = 1$.

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, then the second column, and so on until we have an upper triangular matrix. This is the idea of the Householder QR factorization.

Note: In lecture, we only got as far as the picture in Figure 1; we did not actually write the formula for the Householder reflector that transforms x to something parallel to e_1 (or to a general unit vector y).