Week 8: Wednesday, Oct 14

Teaser

Suppose $A = I - uu^T$ where u is chosen uniformly at random from the Euclidean unit ball (i.e. $\|u\|_2 < 1$). What is the probability that A is diagonally dominant?

Note: I'd consider extra credit for a well-written solution.

Logistics

1. HW 2 is graded. Most did fine, but there are a couple notes:
   (a) If your code doesn't work, please say so. Don't leave me with the impression that you've never even tried to test your codes.
   (b) Some arguments can be tested on the computer. For example, in the last problem set, several of you claimed that $I + uv^T$ is strictly diagonally dominant when $\|u\|_2\|v\|_2 < 1$. This generally is not true, and can be established fairly quickly by testing with random choices of u and v.
   (c) In the last problem, several of you wrote $D = UV^T$ and constructed U and V by incrementally adding columns to an existing matrix. Because of how MATLAB manages memory, that process of incrementally appending columns can be quite slow. It works fine for this example, but it's worth knowing how to avoid that if you want to write MATLAB code that runs as fast as possible.

2. Test drivers for HW 3 have been posted. *I will use these drivers as part of grading.* Please make sure that your codes work with the drivers. Remember, submissions are due Oct 21.

3. I lost last Friday's lecture notes before I had the chance to transcribe them. I'm pretty sure the recovered notes follow the same high-level path that I followed in lecture, but some details may vary. Please do read the notes, though, as there are a few things there that aren't in the book.

Sensitivity revisited

Last time we worked through some rather detailed calculations of the sensitivity of least squares problems under perturbations, and found

$$\frac{\|\delta x\|}{\|x\|} \leq \left(\kappa(A)^2 \tan(\theta) + \kappa(A)\right) \frac{\|\delta A\|}{\|A\|} + \kappa(A) \sec(\theta) \frac{\|\delta b\|}{\|b\|},$$

where θ is the angle between b and the range space of A. The formula looks complicated because it really involves three distinct effects:

1. If b is nearly orthogonal to the range space of A, then small relative changes to b result in large relative changes to the projection of b onto the range space of A. This is why the angle θ plays a role.

2. If we perturb A, then the projection of b onto the perturbed A moves by an amount that depends on $\kappa(A)$ and the angle between b and the range of A.

3. Once we have computed the projection of b onto the range space of A, we still express that projection as Ax and find the coefficients x. A second factor of $\kappa(A)$ comes in from this step.

What if we care not about x, but about $y = Ax$? That is, what if we really want the nearest thing to b in the range space of A, but we don't particularly care about the expression of y in terms of the basis of columns of A? That is, consider

$$y = Ax = A(A^T A)^{-1} A^T b.$$

Note that if $A = QR$ is an "economy" QR decomposition, we have $A^T A = R^T R$ and so

$$y = QR(R^T R)^{-1} R^T Q^T = QQ^T b.$$

The matrix $P = QQ^T$ represents an orthogonal projection:

1. P is a projection:

$$P^2 = (QQ^T)(QQ^T) = Q(Q^T Q)Q^T = QQ^T = P.$$

2. Pb and $(I - P)b$ are always orthogonal:

$$(Pb)^T (I - P)b = b^T P^T (I - P)b = b^T (P(I - P))b = 0.$$

Now, what about the sensitivity of y? This is a messy calculation that I chose not to do in class, but which I will sketch in these notes. The first order formula is

$$\delta y = \delta P\, b + P\, \delta b,$$

where a messy calculation yields

$$\begin{aligned}
\delta P &= (I-P)(\delta A)(A^TA)^{-1}A^T + A(A^TA)^{-1}(\delta A)^T(I-P) \\
&= (I-P)(\delta A)R^{-1}Q^T + QR^{-T}(\delta A)^T(I-P), \\
\delta P\, b &= (I-P)(\delta A)x + QR^{-T}(\delta A)^T r.
\end{aligned}$$

Now we can use $\|(I-P)\| \leq 1$, $\|P\| \leq 1$, and $\|R^{-1}\| = \sigma_n(A)^{-1}$, together with $\|r\|/\|y\| = \tan(\theta)$ and $\|b\|/\|y\| = \sec(\theta)$, to get

$$\begin{aligned}
\frac{\|\delta P\, b\|}{\|y\|} &\leq \left(\frac{\|A\|\|x\|}{\|y\|} + \kappa(A)\tan(\theta) \right) \frac{\|\delta A\|}{\|A\|} \\
\frac{\|P\, \delta b\|}{\|y\|} &\leq \sec(\theta) \frac{\|\delta b\|}{\|b\|}.
\end{aligned}$$

Putting everything together, we have

$$\frac{\|\delta y\|}{\|y\|} \leq \left(\frac{\|A\|\|x\|}{\|y\|} + \kappa(A)\tan(\theta) \right) \frac{\|\delta A\|}{\|A\|} + \sec(\theta)\frac{\|\delta b\|}{\|b\|} \tag{1}$$

$$\leq \kappa(A)\left(1 + \tan(\theta)\right) \frac{\|\delta A\|}{\|A\|} + \sec(\theta)\frac{\|\delta b\|}{\|b\|} \tag{2}$$

Here, $P\,\delta b$ represents the effect of changes in b (part 1 above), and $\delta P\, b$ represents the effect of changes in the range space of A (part 2).

Note that if we multiply (1) by $\sigma_n(A)^{-1}\|y\|/\|x\|$, we have the right hand side of the sensitivity formula for $\|\delta x\|/\|x\|$ (part 3).

The punch line of all this is that the squared condition number appears in our sensitivity formulae only if we care explicitly about x. If x is just an intermediate for computing Ax, we don't care so much.

Sparse least squares and Q-less QR

Suppose we want to solve a full-rank least squares problem in which A is large and sparse. In principal, we could solve the problem via the normal equations

$$A^TAx = A^Tb,$$

or introduce $A = QR$ and multiply $A^T Ax = R^T Rx = b$ by R^{-T} to find

$$Rx = R^{-T} A^T b = Q^T b.$$

Note that there is a very close relation between these approaches; the matrix R in the QR decomposition is a Cholesky factor of $A^T A$ in the normal equations (possibly scaled by a diagonal matrix with ± 1 on the diagonal). As we have discussed, it may not be advantageous to use the normal equations directly, since forming and factoring $A^T A$ brings in the square of the condition number. On the other hand, in a sparse setting it's not necessarily such a good idea to use the usual QR approach, since even if A and $A^T A$ are sparse, Q in general will not be. Consequently, when A is sparse, we would typically use the following steps

1. Possibly permute the columns of A so that the Cholesky factor of $A^T A$ (or the factor R, which has the same structure) remains sparse. See `help colamd` for an example of a generally good permutation.

2. Compute a "Q-less" QR decomposition, e.g. `R = qr(A,0)` in MATLABwhere A is sparse. This does not compute the (usually very dense) Q factor explicitly.

3. Compute $Q^T b$ as $R^{-T}(A^T b)$, since the latter computation involves only a sparse multiply and a sparse triangular solve.

4. Solve $Rx = Q^T b$.

It's not a bad idea to do iterative refinement after this — see `help qr` in MATLAB.

Your homework mission (problem 4) is to introduce these ideas into a constrained least squares code. Note that the permutation is important!