Week 3: Wednesday, Sep 9

Logistics

1. By the end of this week, we will have covered everything necessary to do the homework (due September 21). We will also have covered most of what I wanted from Chapter 1–2.

2. In Problem 4 on the homework, I am *not* looking for a gigantic mess of algebra. What goes wrong in the program is relatively simple, and one can give a terse explanation that tells why the calculation goes awry.

Forward and backward error analysis

At the end of last lecture, we discussed forward and backward error analysis for dot products and matrix-vector multiplies. This analysis is part of a bigger picture. Suppose that we have a space of problem data $\mathcal{P}$ and a space of possible solutions $\mathcal{S}$. We would *like* to evaluate some exact function

$$f : \mathcal{P} \rightarrow \mathcal{S},$$

but because of errors in intermediate steps of our calculation, we really evaluate

$$\hat{f} : \mathcal{P} \rightarrow \mathcal{S}.$$

The forward error way of thinking says that we are given some problem p and the error is the difference $f(p) - \hat{f}(p)$ between the true and the computed solution. The backward error way of thinking says that the $\hat{f}(p)$ we computed is the same as $f(\hat{p})$, an evaluation of the *true* function for some slightly erroneous data. The difference $\hat{p} - p$ is the backward error. If we know something of the sensitivity of f (e.g., if we know a bound on $f'(p)$), then we can estimate the forward error from this backward error.

Orthogonal matrices

To develop fast, stable methods for matrix computation, it will be crucial that we understand different types of structures that matrices can have. This includes both "basis-free" properties, such as orthogonality, singularity, or self-adjointness; and properties such as the location of zero elements that are really associated with a *matrix* rather than with a linear transform.

Orthogonal matrices will be important throughout our work. The usual definition says that square matrix Q is orthogonal if $Q^*Q = I$, but there are other ways to characterize orthogonality as well. For example, a real square matrix Q is orthogonal iff $\|Qv\|_2 = \|v\|_2$ for all v. Why? Recall that for a real vector space,

$$\langle u+v, u+v\rangle = \langle u,u\rangle + \langle v,v\rangle + 2\langle u,v\rangle.$$

With a little algebra, we have

$$\langle u,v\rangle = \frac{1}{2}\left(\|u+v\|_2^2 - \|u\|_2^2 - \|v\|_2^2\right).$$

Therefore, if $\|Qv\|_2 = \|v\|_2$ for every v, we have

$$\begin{aligned}\langle Qu, Qv\rangle &= \frac{1}{2}\left(\|Q(u+v)\|_2^2 - \|Qu\|_2^2 - \|Qv\|_2^2\right) \\ &= \frac{1}{2}\left(\|u+v\|_2^2 - \|u\|_2^2 - \|v\|_2^2\right) = \langle u,v\rangle.\end{aligned}$$

In particular, that means that if e_i denotes the ith column of the identity, then $\langle Qe_i, Qe_j\rangle = \langle e_i, e_j\rangle = \delta_{ij}$, or $Q^*Q = I$.

Because a matrix is orthogonal iff it preserves lengths in the two-norm, we have that

$$\begin{aligned}\|QA\|_2 &= \|A\|_2, & \|AQ\|_2 &= \|A\|_2, \\ \|QA\|_F &= \|A\|_F, & \|AQ\|_F &= \|A\|_F.\end{aligned}$$

There are other important cases of things that remain invariant under orthogonal transformation, too. For example, suppose $Z_1, \ldots, Z_n$ are independent standard normal random variables; then their joint probability density is

$$f(z_1, z_2, \ldots, z_n) = \prod_{i=1}^n \left(\frac{1}{\sqrt{2\pi}} e^{-z_i^2/2}\right) = \frac{e^{-\|z\|_2^2/2}}{(2\pi)^{n/2}}.$$

Because the density depends only on the length of the vector z, we find that $Y = QZ$ has the same density for any orthogonal matrix Q.

Scalar multiples of orthogonal matrices are also the only perfectly conditioned matrices. That is, if $\kappa_2(A) = 1$, then $A = \alpha Q$, where Q is some orthogonal matrix. To see this, recall that

$$\kappa_2(A) = \|A\|_2 \|A^{-1}\|_2 = \frac{\max_{\|v\|_2=1} \|Av\|_2}{\min_{\|u\|_2=1} \|Au\|_2},$$

so if $\kappa_2(A) = 1$, the images of all unit vectors under A have the same length – which means that the lengths of all vectors are scaled by the same amount by the action of A. Define $\alpha = \|Av\|/\|v\|$ to be the scaling factor; then $Q = \alpha^{-1}A$ scales the length of every vector by one, which means that Q is orthogonal.

The singular value decomposition

The fact that orthogonal transforms leave so many metric properties of matrices unchanged suggests the following: find orthogonal transformations that, when applied to a matrix A, result in a matrix that is as structurally simple as possible. The result of this is the singular value decomposition (SVD), which is discussed in 2.5.3–2.5.5 in the book (a discussion we mostly followed in class). That is, we can write

$$A = U\Sigma V^*$$

where U and V^* are orthogonal matrices and Σ is a diagonal matrix with non-negative diagonal entries that — according to convention — appear in descending order.

The rank of a matrix A is given by the number of nonzero singular values. In computational practice, we would say that a matrix has numerical rank k if exactly k singular values are sufficiently greater than zero. The rank of a matrix is theoretically interesting and useful, but it is also computationally useful to realize when a matrix is low rank, because that low rank structure can be used for fast multiplication. Suppose the SVD for $A \in \mathbb{R}^{n\times n}$ is

$$A = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1 & V_2 \end{bmatrix}^*$$

where $U_1, V_1 \in \mathbb{R}^{n\times k}$ and $\Sigma_1 \in \mathbb{R}^{k\times k}$. If we don't use anything about the structure of A, then we take $O(n^2)$ time to compute $y = Ax$. If we write

$y = U_1(\Sigma_1(V_1^* x))$, then it takes $O(nk)$ time to compute y. If $k \ll n$, this may be a substantial savings! So there is a potential efficiency win in recognizing when a matrix has low rank, particularly when the matrix can be written as an outer product from the outset so that we don't have to compute an SVD or similar factorization.