2023-02-15

1 Perturbation theory

Previously, we described a general error analysis strategy: derive forward error bounds by combining a sensitivity estimate (in terms of a *condition number*) with a *backward* error analysis that explains the computed result as the exact answer to a slightly erroneous problem. To follow that strategy here, we need the sensitivity analysis of solving linear systems.

Suppose that $Ax = b$ and that $\hat{A}\hat{x} = \hat{b}$, where $\hat{A} = A + \delta A$, $\hat{b} = b + \delta b$, and $\hat{x} = x + \delta x$. Then

$$\delta A\, x + A\, \delta x + \delta A\, \delta x = \delta b.$$

Assuming the delta terms are small, we have the linear approximation

$$\delta A\, x + A\, \delta x \approx \delta b.$$

We can use this to get δx alone:

$$\delta x \approx A^{-1}(\delta b - \delta A\, x);$$

and taking norms gives us

$$\|\delta x\| \lesssim \|A^{-1}\|(\|\delta b\| + \|\delta A\|\|x\|).$$

Now, divide through by $\|x\|$ to get the relative error in x:

$$\frac{\|\delta x\|}{\|x\|} \lesssim \|A\|\|A^{-1}\| \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|A\|\|x\|} \right).$$

Recall that $\|b\| \leq \|A\|\|x\|$ to arrive at

$$\frac{\|\delta x\|}{\|x\|} \lesssim \kappa(A) \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|} \right),$$

where $\kappa(A) = \|A\|\|A^{-1}\|$. That is, the relative error in x is (to first order) bounded by the condition number times the relative errors in A and b. We can go beyond first order using Neumann series bounds – but perhaps not today.

2 Backward error in Gaussian elimination

Solving $Ax = b$ in finite precision using Gaussian elimination followed by forward and backward substitution yields a computed solution $\hat{x}$ *exactly* satisfies

$$(A + \delta A)\hat{x} = b, \tag{1}$$

where $|\delta A| \lesssim 3n\epsilon_{\text{mach}}|\hat{L}||\hat{U}|$, assuming $\hat{L}$ and $\hat{U}$ are the computed L and U factors.

I will now briefly sketch a part of the error analysis following Demmel's treatment (§2.4.2, *Applied Numerical Linear Algebra*). Here is the idea. Suppose $\hat{L}$ and $\hat{U}$ are the computed L and U factors. We obtain $\hat{u}_{jk}$ by repeatedly subtracting $l_{ji}u_{ik}$ from the original a_{jk}, i.e.

$$\hat{u}_{jk} = \text{fl}\left(a_{jk} - \sum_{i=1}^{j-1} \hat{l}_{ji}\hat{u}_{ik}\right).$$

Regardless of the order of the sum, we get an error that looks like

$$\hat{u}_{jk} = a_{jk}(1 + \delta_0) - \sum_{i=1}^{j-1} \hat{l}_{ji}\hat{u}_{ik}(1 + \delta_i) + O(\epsilon_{\text{mach}}^2)$$

where $|\delta_i| \leq (j-1)\epsilon_{\text{mach}}$. Turning this around gives

$$\begin{aligned} a_{jk} &= \frac{1}{1+\delta_0}\left(\hat{l}_{jj}\hat{u}_{jk} + \sum_{i=1}^{j-1} \hat{l}_{ji}\hat{u}_{ik}(1+\delta_i)\right) + O(\epsilon_{\text{mach}}^2) \\ &= \hat{l}_{jj}\hat{u}_{jk}(1-\delta_0) + \sum_{i=1}^{j-1} \hat{l}_{ji}\hat{u}_{ik}(1+\delta_i-\delta_0) + O(\epsilon_{\text{mach}}^2) \\ &= \left(\hat{L}\hat{U}\right)_{jk} + E_{jk}, \end{aligned}$$

where

$$E_{jk} = -\hat{l}_{jj}\hat{u}_{jk}\delta_0 + \sum_{i=1}^{j-1} \hat{l}_{ji}\hat{u}_{ik}(\delta_i - \delta_0) + O(\epsilon_{\text{mach}}^2)$$

is bounded in magnitude by $(j-1)\epsilon_{\text{mach}}(|L||U|)_{jk} + O(\epsilon_{\text{mach}}^2)$[1]. A similar argument for the components of $\hat{L}$ yields

$$A = \hat{L}\hat{U} + E, \text{ where } |E| \leq n\epsilon_{\text{mach}}|\hat{L}||\hat{U}| + O(\epsilon_{\text{mach}}^2).$$

In addition to the backward error due to the computation of the LU factors, there is also backward error in the forward and backward substitution phases, which gives the overall bound (1).

3 Pivoting

The backward error analysis in the previous section is not completely satisfactory, since $|L||U|$ may be much larger than $|A|$, yielding a large backward error overall. For example, consider the matrix

$$A = \begin{bmatrix} \delta & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \delta^{-1} & 1 \end{bmatrix} \begin{bmatrix} \delta & 1 \\ 0 & 1-\delta^{-1} \end{bmatrix}.$$

If $0 < \delta \ll 1$ then $\|L\|_\infty \|U\|_\infty \approx \delta^{-2}$, even though $\|A\|_\infty \approx 2$. The problem is that we ended up subtracting a huge multiple of the first row from the second row because δ is close to zero — that is, the leading principle minor is *nearly* singular. If δ were exactly zero, then the factorization would fall apart even in exact arithmetic. The solution to the woes of singular and near singular minors is pivoting; instead of solving a system with A, we re-order the equations to get

$$\hat{A} = \begin{bmatrix} 1 & 1 \\ \delta & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \delta & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1-\delta \end{bmatrix}.$$

Now the triangular factors for the re-ordered system matrix $\hat{A}$ have very modest norms, and so we are happy. If we think of the re-ordering as the effect of a permutation matrix P, we can write

$$A = \begin{bmatrix} \delta & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \delta & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1-\delta \end{bmatrix} = P^T LU.$$

[1] It's obvious that $E_j k$ is bounded in magnitude by $2(j-1)\epsilon_{\text{mach}}(|L||U|)_{jk} + O(\epsilon_{\text{mach}}^2)$. We cut a factor of two if we go down to the level of looking at the individual rounding errors during the dot product, because some of those errors cancel.

Note that this is equivalent to writing $PA = LU$ where P is another permutation (which undoes the action of P^T).

If we wish to control the multipliers, it's natural to choose the permutation P so that each of the multipliers is at most one. This standard choice leads to the following algorithm:

```
# Return L, U, p s.t. A[p,:] = L*U and the largest entry of L has magnitude 1
function my_pivoted_lu(A)

  n = size(A)[1]
  A = copy(A)      # Make a local copy to overwrite
  piv = zeros(Int, n) # Space for the pivot vector
  piv[1:n] = 1:n

  for j = 1:n-1

    # Find ipiv >= j to maximize |A(i,j)|
    ipiv = (j-1)+findmax(abs.(A[j:n,j]))[2]

    # Swap row ipiv and row j and record the pivot row
    A[ipiv,:], A[j,:] = A[j,:], A[ipiv,:]
    piv[ipiv], piv[j] = piv[j], piv[ipiv]

    # Compute multipliers and update trailing submatrix
    A[j+1:n,j] /= A[j,j]
    A[j+1:n,j+1:n] -= A[j+1:n,j]*A[j,j+1:n]'

  end

  UnitLowerTriangular(A), UpperTriangular(A), piv
end
```

By design, this algorithm produces an L factor in which all the elements are bounded by one. But what about the U factor? There exist pathological matrices for which the elements of U grow exponentially with n. But these examples are extremely uncommon in practice, and so, in general, Gaussian elimination with partial pivoting does indeed have a small backward error. Of course, the pivot growth is something that we can monitor, so in the unlikely event that it *does* look like things are blowing up, we can tell there is a problem and try something different.

When problems do occur, it is more frequently the result of ill-conditioning in the problem than of pivot growth during the factorization.