18.409 The Behavior of Algorithms in Practice 28 Feb. 2002

Lecture 6

Lecturer: Dan Spielman *Scribe: Brian Sutton*

1 Outline

- When Gaussian elimination with partial pivoting fails.
- Analysis of complete pivoting.
- Speeding up the solution of linear systems.

2 When partial pivoting fails

2.1 An example

In lecture 3, we saw an example of a linear system that Matlab fails to solve. Here is a similar system:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 1 \\ -1 & -1 & 1 & 0 & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{pmatrix} x = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

We will solve this system using Gaussian elimination with partial pivoting.

First, we permute the rows so that the $(1,1)$-entry has the largest magnitude in the first column. Actually, we do not need to permute; our pivot is already in the right place.

Eliminating the nonzero entries in the first column, we get

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 2 \\ 0 & -1 & 1 & 0 & 2 \\ 0 & -1 & -1 & 1 & 2 \\ 0 & -1 & -1 & -1 & 2 \end{pmatrix}.$$

The rest of the reduction proceeds similarly, obtaining

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & 4 \\ 0 & 0 & 0 & 1 & 8 \\ 0 & 0 & 0 & 0 & 16 \end{pmatrix}.$$

The exponentially large magnitude of the lower-right entry is an artifact of the reduction. Because of rounding, the lower-right entry of the original matrix may not be recoverable from the reduced matrix. (Technically, Matlab solves this system, because every entry in the last column is exactly a power of two. See the file `kahan2.m` for a very similar system that Matlab cannot solve.)

2.2 Smoothed analysis

What does smoothed analysis tell us about the example? On one hand, we have already seen that if an entire matrix has noise in it, then Gaussian elimination requires relatively few bits of precision, even without pivoting. But on the other hand, if we perturb only the nonzero entries in the above system, then the error still creeps in.

Hence, our current application of smoothed analysis to Gaussian elimination fails for this example. Applying smoothed analysis to Gaussian elimination with *complete* pivoting may resolve this issue.

3 Complete pivoting

Gaussian elimination with complete pivoting permutes rows *and* columns to ensure that the pivot is the largest magnitude entry in the entire submatrix that remains to be row reduced.

It is easy to check that complete pivoting guarantees $\|L\|_\infty < 1$. The algorithm also gives

Theorem 1 (Wilkinson).

$$\frac{\|U\|_\infty}{\|A\|_\infty} \leq n^{\frac{1}{4}((\log n)+1)}.$$

Therefore, complete pivoting requires no more than $\log^2 n$ *bits of precision for best possible accuracy.*

However, people do not use complete pivoting in practice. Partial pivoting rarely fails, and complete pivoting requires twice as many floating point operations to find the largest magnitude entry (the pivot).

We should also note that it may be possible to improve upon Wilkinson's theorem. Nearly always, the growth factor is actually less than n. Perhaps by assuming that there is noise in the input, we can find a better bound. Or perhaps we can calculate a better bound directly.

Proof of Wilkinson. Let $A^{(r)}$ denote the lower-right r-by-r submatrix that is obtained after $n-r-1$ eliminations. Assume that complete pivoting has already been performed, so that the largest magnitude entry in $A^{(r)}$ is the upper-left entry. Let p_r denote this entry, i.e. the pivot.

Then $\|A\|_\infty = p_n$ and $\|A^{(r)}\|_\infty = p_r$, so we need to bound $|p_r|/|p_n|$. (Note that $\frac{\|U\|_\infty}{\|A\|_\infty} \leq \max_r \frac{\|A^{(r)}\|_\infty}{\|A\|_\infty}$.)

Wilkinson's theorem will be a consequence of the next theorem. □

Theorem 2.

$$\frac{|p_1|}{|p_n|} \leq 2^{\frac{1}{2}\log_2\left(2\cdot 3^{1/2}\cdot 4^{1/3}\cdot\dots\cdot n^{1/(n-1)}\right)+\frac{1}{2}\log n}.$$

Proof. The result follows from two facts:

$$\det(A^{(r)}) = \prod_{i=1}^{r} p_i, \tag{1}$$

$$\det(A^{(r)}) \leq (\sqrt{r}\,|p_r|)^r. \tag{2}$$

(Inequality (2) is Hadamard's inequality.) Immediately, we have

$$\prod_{i=1}^{r} p_i \leq (\sqrt{r}\,|p_r|)^r$$

If $q_r = \log|p_r|$, then $\sum_{i=1}^{r} q_i \leq \frac{r}{2}\log r + r q_r$, so

$$\sum_{i=1}^{r-1} q_i \leq \frac{r}{2}\log r + (r-1)q_r, \tag{3}$$

which implies

$$\sum_{i=1}^{r-1} \frac{q_i}{r(r-1)} \leq \frac{\log r}{2(r-1)} + \frac{q_r}{r} = \frac{1}{2}\log\left(r^{\frac{1}{r-1}}\right) + \frac{q_r}{r}. \tag{4}$$

Now, rewriting (1), we obtain

$$\frac{\sum_{i=1}^{n} q_i}{n-1} = \frac{\log(\det(A))}{n-1}. \tag{5}$$

Summing inequality (4) over $r = 2, \ldots, n-1$ with (5), we get

$$\sum_{r=2}^{n} \frac{q_{r-1}}{r-1} + \frac{q_n}{n-1} \leq \frac{1}{2}\log\left(2 \cdot 3^{1/2} \cdot 4^{1/3} \cdot \cdots \cdot (n-1)^{1/(n-2)}\right) + \sum_{r=2}^{n-1} \frac{q_r}{r} + \frac{\log(\det(A))}{n-1},$$

so

$$q_1 + \frac{q_n}{n-1} \leq \frac{1}{2}\log\left(2 \cdot 3^{1/2} \cdot 4^{1/3} \cdot \cdots \cdot (n-1)^{1/(n-2)}\right) + \frac{\log(\det(A))}{n-1}.$$

Using (1) and (3), we can bound the last term in the RHS:

$$\frac{\log(\det(A))}{n-1} = \frac{1}{n-1}\sum_{i=1}^{n} q_i \leq \frac{n}{2(n-1)}\log n + q_n.$$

Thus,

$$q_1 + \frac{q_n}{n-1} \leq \frac{1}{2}\log\left(2 \cdot 3^{1/2} \cdot 4^{1/3} \cdot \cdots \cdot (n-1)^{1/(n-2)}\right) + \frac{n}{2(n-1)}\log n + q_n,$$

so

$$\begin{aligned} q_1 - q_n &\leq q_1 - \frac{q_n(n-2)}{n-1} \\ &\leq \frac{1}{2}\log\left(2 \cdot 3^{1/2} \cdot 4^{1/3} \cdot \cdots \cdot (n-1)^{1/(n-2)}\right) + \frac{1}{2}\log n^{1/(n-1)} + \frac{n-1}{2(n-1)}\log n, \end{aligned}$$

and finally,

$$q_1 - q_n \leq \frac{1}{2}\log\left(2 \cdot 3^{1/2} \cdot 4^{1/3} \cdot \cdots \cdot (n-1)^{1/(n-2)} \cdot n^{1/(n-1)}\right) + \frac{1}{2}\log n. \ \square$$

4 Speeding up the solution of linear systems

If a matrix $A = (a_{ij})$ is banded within b diagonals of the main diagonal, i.e. $a_{ij} \neq 0$ implies $|i-j| \leq b-1$, then Gaussian elimination with partial pivoting can reduce the matrix in time $O(nb^2)$ and space $O(nb)$. (You can check that the upper-triangular matrix U in the factorization is banded by $2b$ (approximately).)

Given an arbitrary matrix, we would like to permute the rows and columns so that the matrix is banded. If the matrix is symmetric, then this problem becomes a graph problem. Define the graph of a symmetric n-by-n $A = (a_{ij})$ to be an undirected graph on n vertices in which there is an edge (i, j) if and only if $a_{ij} \neq 0$. (The diagonal of A is irrelevant.)

4.1 Breadth-first search

4.1.1 Cuthill-McKee '69

One of the first algorithms to transform matrices into banded matrices was developed by Cuthill and McKee in 1969. Given an arbitrary vertex v, the algorithm is as folows:

1. Perform a breadth-first search on the graph of A rooted at v. Label each vertex with its distance from v, so that the vertices are partitioned into distinct "levels."
2. Order the vertices by level, i.e. find a permutation σ such that if $\sigma(i) < \sigma(j)$, then the level of i is no higher than the level of j.
3. Permute the rows and columns of A according to the permutation σ.

Note that if the BFS partitions the vertices so that each level contains relatively few vertices, then the resulting matrix is banded into a relatively small region. To be precise, if L_i is the set of vertices in level i, then the bandwidth of the permuted matrix is $\leq 3 \cdot \max_i |L_i|$.

4.1.2 Turner '86

In 1986, Turner analyzed Cuthill-McKee, finding an improvement in the process. His improvement was

1. Start with a random root v.
2. Run Cuthill-McKee.
3. Let w be the last vertex (corresponding to row and column n).
4. Run Cuthill-McKee from w.

Turner's analysis was probablistic. For a fixed b, he constructed a random (0, 1)-matrix in which each entry within b diagonals from the main diagonal was 1 with probability p and all other entries were 0. He found that for $p \geq \frac{\log n}{n}$, the bandwidth of the matrix was almost certainly b. Cuthill-McKee produced a permuted matrix with bandwith about $3b$. In contrast, Turner's improvement achieved a bandwith of $b + O(\log n)$.

4.1.3 Feige-Krauthgamer '98

Feige and Krauthgamer performed a more general analysis. Starting with an arbitrary (0, 1)-matrix of bandwidth b, they flipped entries in the band with probability p. (The case in which the original matrix is the zero matrix is Turner's analysis.)

Feige and Krauthgamer's analysis deals with a certain class of perturbed matrices, similar to smoothed complexity analysis.

4.2 Graph Separators

For a graph G, an α-vertex separator is a subset $C \subseteq V$ such that $V \setminus C$ is the disjoint union of sets A and B with no edges between A and B, and $|A|, |B| \leq \alpha|V|$.

If we can find an α-vertex separator, then we can make progress toward rearranging the matrix into banded form, since there are no edges between the vertices in A and the vertices in B.

Theorem 3 (Lipton-Tarjan). *Every n-node planar graph has a $\frac{2}{3}$-vertex separator C satisfying $|C| \leq \sqrt{8n}$.*

Definition 4. *If S is a family of graphs closed under subgraph, then S satisfies an $f(n)$-separator theorem if there exists an $\alpha < 1$ such that all graphs in S have α-separators of size $\leq f(|V|)$.*

Theorem 5 (Lipton-Rose-Tarjan). *If S has cn^γ-separators, then if G is the non-zero structure of a linear system, and $G \in S$, then we can solve G in space and time given by the following table.*

	space = fill	*time*
$\gamma = \frac{1}{2}$	$O(n \log n)$	$O(n^{3/2})$
$\gamma > \frac{1}{2}$	$n^{2\gamma}$	$n^{3\gamma}$

4.3 Non-planar graphs

What if the graph of a matrix is not planar? There are a number of heuristic contenders. One of the first was developed by Kernighan and Cuthill.

A newer, promising approach uses the eigenvector structure of the Laplacian matrix of the graph of A. Each diagonal entry of a Laplacian matrix of a graph is the degree of the corresponding vertex. The off-diagonal entries are negated edge weights.

The Laplacian matrix is positive semidefinite, with 0 as an eigenvalue with multiplicity 1 (if the graph is connected). The eigenvector corresponding to the second smallest eigenvalue can be used to order the vertices. The ordering can be used to cut the graph efficiently. Details will be provided in the next lecture.