Gaussian elimination without rounding

From László Lovász's lecture notes (by Péter Gács)

February 15, 2018

The determinant

Let $A = (a_{ij})$ be an arbitrary $n \times n$ matrix consisting of integers.

Let us understand, first of all, that the polynomial computation of $\det(A)$ is not inherently impossible, i.e. the result can be written with polynomially many digits. Let $K = \max |a_{ij}|$, then to write A we need obviously at least $n^2 \log K$ bits. On the other hand, the definition of determinants gives

$$|\det(A)| \leq n! K^n,$$

hence $\det(A)$ can be written using

$$\log(n! K^n) + O(1) \leq n(\log n + \log K) + O(1)$$

bits. Thus, $\det(A)$ can be written with polynomially many bits. Linear algebra gives a formula for each element of $\det(A^{-1})$ as the quotient of two subdeterminants of A. This shows that A^{-1} can also be written with polynomially many digits.

Exercise 1 Show that if A is a square matrix consisting of integers then to write $\det(A)$ we need at most as many bits as to write A. [Hint: If $a_1, \ldots, a_n$ are the row vectors of A then $|\det(A)| \leq |a_1| \cdots |a_n|$ (this so-called "Hadamard-inequality" is analogous to the statement that the area of a parallelogram is smaller than the product of the lengths of its sides).]

Gaussian elimination.

The usual procedure to compute the determinant is the so-called Gaussian elimination. We can view this as the transformation of the matrix into a lower triangular matrix with column operations. These transformations do not change the determinant but in the triangular matrix, the computation of the determinant is

more convenient: we must only multiply the diagonal elements to obtain it. (It is also possible to obtain the inverse matrix from this form; we will not deal with this separately.)

Suppose that for all i such that $1 \leq i \leq t$ we have achieved already that in the ith row, only the first i positions hold a nonzero element. Pick a nonzero element from the last $n - t$ columns (if there is no such element we stop). We call this element the *pivot element* of this stage. Let us rearrange the rows and columns so that this element gets into position $(t + 1, t + 1)$. Subtract column $t + 1$, multiplied by $a_{t+1,i}/a_{t+1,t+1}$, from column i for all $i = t + 2, \ldots, n$, in order to get 0's in the elements $(t+1, t+2), \ldots, (t+1, n)$. It is known that the subtractions do not change value of the determinant and the rearrangement (involving as many exchanges of rows as of columns) also does not change the determinant.

Since one iteration of the Gaussian elimination uses $O(n^2)$ arithmetical operations and n iterations must be performed this means $O(n^3)$ arithmetical operations. But the problem is that we must also divide, and not with remainder. This does not cause a problem over a finite field but it does in case of the rational field. We assumed that the elements of the original matrix are integers; but during the running of the algorithm, matrices also occur that consist of rational numbers. In what form should these matrix elements be stored? The natural answer is that as pairs of integers (whose quotient is the rational number).

But do we require that the fractions be in reduced form, that is that their numerator and denominator be relatively prime to each other? We could do this but then we have to reduce each matrix element after each iteration, for which we would have to perform the Euclidean algorithm. This can be performed in polynomial time but it is a lot of extra work, desirable to avoid. (Of course, we also have to show that in the reduced form, the occurring numerators and denominators have only polynomially many digits.)

We could also choose not to require that the matrix elements be in reduced form. Then we define the sum and product of two rational numbers a/b and c/d by the following formulas: $(ad + bc)/(bd)$ and $(ac)/(bd)$. With this convention, the problem is that the numerators and denominators occurring in the course of the algorithm can be very large (have a nonpolynomial number of digits)!

Fortunately, we can give a procedure that stores the fractions in partially reduced form, and avoids both the reduction and the excessive growth of the number of digits. For this, let us analyze a little the matrices occurring during Gaussian elimination. We can assume that the pivot elements are, as they come, in positions $(1, 1), \ldots, (n, n)$, i.e., we do not have to permute the rows and columns. Let $(a_{ij}^{(k)})$ $(1 \leq i, j \leq n)$ be the matrix obtained after k iterations. Let us denote the elements in the main diagonal of the final matrix, for simplicity, by $d_1, \ldots, d_n$

(thus, $d_i = a_{ii}^{(n)}$). Let $D^{(k)}$ denote the submatrix determined by the first k rows and columns of matrix A, and let $D_{ij}^{(k)}$, for $k+1 \le i, j \le n$, denote the submatrix determined by the first k rows and the ith row and the first k columns and the jth column. Let $d_{ij}^{(k)} = \det(D_{ij}^{(k)})$. Obviously, $\det(D^{(k)}) = d_{kk}^{(k-1)}$.

Lemma 1

$$a_{ij}^{(k)} = \frac{d_{ij}^{(k)}}{\det(D^{(k)})}.$$

Proof. If we compute $\det(D_{ij}^{(k)})$ using Gaussian elimination, then in its main diagonal, we obtain the elements $d_1, \ldots, d_{k+1}, a_{ij}^{(k)}$. Thus

$$d_{ij}^{(k)} = d_1 \cdots d_{k+1} \cdot a_{ij}^{(k)} \text{ and similarly } \det(D^{(k)}) = d_1 \cdots d_{k+1}.$$

Dividing these two equations by each other, we obtain the lemma. □

By this lemma, every number occurring in the Gaussian elimination can be represented as a fraction both the numerator and the denominator of which is a determinant of some submatrix of the original A matrix. In this way, a polynomial number of digits is certainly enough to represent all the fractions obtained.

However, it is not necessary to reduce all fractions obtained in the process. By the definition of Gaussian elimination we have that

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - \frac{a_{i,k+1}^{(k)} a_{k+1,j}^{(k)}}{a_{k+1,k+1}^{(k)}}$$

and hence

$$d_{ij}^{(k+1)} = \frac{d_{ij}^{(k)} d_{k+1,k+1}^{(k)} - d_{i,k+1}^{(k)} d_{k+1,j}^{(k)}}{d_{k,k}^{(k-1)}}.$$

This formula can be considered a recursion for computing the numbers $d_{ij}^{(k)}$. Since the left-hand side is integer, the division can be carried out exactly. Using the above considerations, we find that the number of digits in the quotient is polynomial in terms of the size of the input.

Other possibilities

Here are some alternatives for exact computation.

Approximation We can approximate the number by binary "decimals" of limited accuracy (as it seems natural from the point of view of computer implementation), allowing, say, p binary digits after the binary "decimal point". Then the result is only an approximation, but since the determinant is an integer, it would be enough to compute it with an error smaller than 1/2. Using the methods of numerical analysis, it can be found out how large must p be chosen to make the error in the end result smaller than 1/2. It turns out that a polynomial number of digits is enough (see [1]) and this leads to a polynomial algorithm.

Modular computation This possibility is probably the most practical one. If $m > |\det(A)|$ then it is enough to determine the value of $\det(A)$ modulo m. If m is a prime number then computing modulo m, we don't have to use fractions. Since we know that $|\det(A)| < n!K^n$ it would be enough to choose for m a prime number greater than $n!K^n$. This is, however, not easy (see the section of these notes on randomized algorithms), hence we can choose m as the product of different small primes: $m = 2 \cdot 3 \cdots p_{k+1}$ where for k we can choose, e.g., the number of all digits occurring in the representation of A. Then it is easy to compute the remainder of $\det(A)$ modulo p_i for all p_i using Gaussian elimination in the field of residue classes, and then we can compute the remainder of $\det(A)$ modulo m using the Chinese Remainder Theorem. Since k is small (and it is known that there are sufficiently many small primes) we can afford to find the first k primes simply by the sieve method and still keep the algorithm polynomial. But the cost of this computation must be judged differently anyway since the same primes can then be used for the computation of arbitrarily many determinants.

The modular method is successfully applicable in a number of other cases. We can consider it as a coding of the integers in a way different from the binary (or decimal) number system: we code the integer n by its remainder after division by the primes 2,3, etc. This is an infinite number of bits but if we know in advance that no number occurring in the computation is larger than N then it is enough to consider the first k primes whose product is larger than N. In this coding, the arithmetic operations can be performed very simply, and even in parallel for the different primes. Comparison by magnitude is, however, awkward.

Bibliography

[1] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford, 1965.