CS761 Derandomization and Pseudorandomness **2022-23 Sem I**

Lecture 1: 01-08-2022

Scribe: Amit Rajaraman *Lecturer: Rohit Gurjar*

Problem. Given three $n \times n$ matrices A, B, C, decide whether $AB = C$.

One naïve way to do this is to compute AB and check if it is identical to C. The naïve implementation of this runs in $O(n^3)$, while the best known implementation at the time runs in about $O(n^{2.373\cdots})$.
Can we do the required in $O(n^2)$ time, perhaps in a random fashion (with some probability of failure)?

Consider the following algorithm to start with. For each row in C, choose an entry randomly and verify that it matches the corresponding entry in AB. In a similar spirit, a second algorithm is to choose n entries of C randomly and verify.

If $AB = C$, it is clear that no matter how we choose to test, we shall return that the two are indeed equal. The probability we would like to minimize is

$$\Pr[\text{the algorithm outputs yes} \mid AB \neq C].$$

Of course, this probability depends on A, B, C. The non-deterministic part of it is the randomness inherent in the algorithm, not in some choosing of A, B, C.

When AB and C differ at only one entry, the earlier proposed algorithm has a success probability of $1/n$ (so the quantity mentioned above is $1 - 1/n$). This is very bad, as it means that to reduce the failure probability to some constant, we would need to repeat this $\Omega(n)$ times.

An algorithm that does the job is as follows.
Randomly choose $r \in \{0,1\}^n$. Compute ABr and Cr, and verify that the two are equal. This is an $O(n^2)$ algorithm, since multiplying a matrix with a vector takes $O(n^2)$ and we perform this operation thrice, in addition to an $O(n)$ verification step at the end.

We claim that the failure probability of this algorithm is at most $1/2$.
The failure probability can be rephrased as follows. Let $x, y \in \mathbb{R}^{1\times n}$. What is $\Pr[xr = yr \mid x \neq y]$? The earlier failure probability is at most equal to this, with equality attained (in a sense) when the two matrices differ at exactly one row.
This in turn is equivalent to the following. Let $z \in \mathbb{R}^{1\times n}$. What is $\Pr[zr = 0 \mid z \neq 0]$? Suppose that $z_i \neq 0$ for some i. For any choice of the remaining $n-1$ bits, at most one of the two options for the ith bit can result in $zr = 0$.
Let us do this slightly more formally. Assume without loss of generality that $z_n \neq 0$. Then,

$$\begin{aligned}\Pr\left[z_1r_1 + \cdots + z_nr_n = 0 \mid z_n \neq 0\right] &= \Pr\left[r_n = -\frac{z_1r_1 + \cdots + z_{n-1}r_{n-1}}{z_n} \mid z_n \neq 0\right] \\ &\leq \max_{r_1,\ldots,r_{n-1}} \Pr\left[r_n = -\frac{z_1r_1 + \cdots + z_{n-1}r_{n-1}}{z_n} \mid z_n \neq 0, r_1, \ldots, r_{n-1}\right]\end{aligned}$$

which is plainly at most $1/2$ – we cannot have that both 0 and 1 are equal to the quantity of interest!
Observe that if we instead choose r from $\{0, 1, \ldots, q-1\}^n$ instead, the failure probability now goes down at most $1/q$. There is a tradeoff at play here between the reduction in the failure probability and the increase in the number of random bits, which goes from n to $O(n \log q)$.

Can we reduce the number of random bits in this algorithm? Can we make it deterministic?

To answer the question of determinism, suppose the algorithm designer chooses k vectors $r^{(1)}, \ldots, r^{(k)} \in \mathbb{R}^n$ and tests whether $ABr^{(i)} = Cr^{(i)}$. This will fail if $k < n$. Indeed, an adversarial input is a z that is nonzero but with $zr^{(i)} = 0$ for $1 \leq i \leq k$.
The determinism here is in the sense that the vectors are chosen before the inputs are provided.

On the other hand, we *can* reduce the number of random bits used. In fact, we can go to about $O(\log n)$ bits.
The goal of derandomization is to use a smaller number of random bits, perhaps by conditioning together previously independent bits, without losing the power of the earlier independent bits.
Let

$$A(x) = a_0 + a_1 x + \cdots + a_d x^d$$

be a nonzero polynomial of degree d. Choose x randomly from $\{0, 1, \ldots, q-1\}$. It is not difficult to see that

$$\Pr_{x \sim \{0,1,\ldots,q-1\}} [A(x) = 0] \leq \frac{d}{q}$$

by the Fundamental Theorem of Algebra.
Inspired by this, we can reduce randomness as follows. Choose x randomly from $\{0, 1, \ldots, 2n-1\}$, and set $r = (1, x, x^2, \ldots, x^{n-1})$. Then,

$$\Pr[z_1 r_1 + z_2 r_2 + \cdots + z_n r_n = 0] = \Pr\left[z_1 + z_2 x + z_2 x^2 + \cdots + z_n x^n\right] \leq \frac{n-1}{2n-1} \leq \frac{1}{2}.$$

There are some other issues that enter the picture here, namely the bit complexity now that x^{n-1} has $O(n)$ bits. One easy fix for this is to perform all operations modulo some prime.