Randomized Algorithms
Lecture 2

Kousha Etessami

School of Informatics
University of Edinburgh

Prior lecture: checking polynomial identities

In the first lecture we considered the problem of taking two polynomials of degree d, $F(x)$ written as a product of degree-1 polynomials, and $G(x)$ as a standard sum of monomial terms, and deciding whether or not $F(x)$ is identical to $G(x)$.

The basic algorithm takes a single uniform random sample x_1 from the set $\{1, \ldots, 100d\}$ and calculates whether $F(x_1)$ and $G(x_1)$ are equal. This testing algorithm gives an incorrect answer with probability at most $\frac{1}{100}$ ("one-sided" error).

- The sample drawn to perform the test is just a single value chosen uniformly from $\{1, \ldots, 100d\}$. An easy probability distribution to understand.
- To refine the algorithm, we can do k trials (and answer "No" if we ever get "No" in any trial; answer "Yes" otherwise.). This "powers up" the error probability, reducing it to at most $\frac{1}{100^k}$.

Matrix multiplication verification

We are given three $n \times n$ matrices, A, B, & C, and we are asked to verify whether or not

$$AB \stackrel{?}{=} C,$$

without carrying out the costly task of multiplying out AB.

Recall that the "obvious" algorithm for evaluating AB would take $\Theta(n^3)$ time steps (arithmetic operations). The algorithm with the current best known asymptotic upper bound takes $O(n^{2.37286\cdots})$ steps ([Alman-Vassilevska Williams,2021]). But these are very involved algorithms, building on decades of prior work (starting with [Strassen'69]), and they involve rather large constants hidden in the big-O notation.

We will instead show how to verify the identify $AB = C$ (with high probability) in $O(n^2)$ time, using a very simple and easy randomized algorithm.

Matrix multiplication verification

Assume that the entries in the matrices are integers in $\mathbb{Z}$, or even rational numbers in $\mathbb{Q}$.

The algorithm is parametrized by some natural number $k \geq 1$. The larger k is, the smaller the probability of failure, but also the larger the running time.

Algorithm MMVERIFY(n, A, B, C, k)

1. **for** $j = 1, \ldots, k$ **do**
2. Generate a vector $x \in \{0, 1\}^n$ uniformly at random.
3. Calculate vector $y^B = B \cdot x$ in $O(n^2)$ time.
4. Calculate vector $y^{AB} = A \cdot y^B$ in $O(n^2)$ time.
5. Calculate vector $y^C = C \cdot x$ in $O(n^2)$ time.
6. **if** $y^{AB} \neq y^C$ (i.e., if they differ in *any* coordinate)
7. **return** "NO"
8. **return** "YES"

Analysing MMVerify

First, let us observe that each of steps 3., 4., 5. can be carried out in $O(n^2)$ steps, for a given vector $x \in \{0, 1\}^n$.
Next, for the analysis, we will show:

"One-sided error"

- if $AB = C$: In this case, we know that $AB \cdot x = Cx$ for every $x \in \{0, 1\}^n$. Hence MMVERIFY is guaranteed to return the correct answer "YES".
- if $AB \neq C$: We will next show that in this case, when a vector x is drawn u.a.r. from $\{0, 1\}^n$, the probability that $AB \cdot x = C \cdot x$ is at most $1/2$.

After this analysis, we will calculate the effect of doing k trials.

Analysing MMVerify: $AB \neq C$

Consider the two $n \times n$ matrices AB and C. We are assuming they are not identical, so there must be *at least* one cell (i^*, j^*) such that the values $(AB)_{i^* j^*} \neq C_{i^* j^*}$.

Let $D = (AB - C)$. Then equivalently, we have $D_{i^* j^*} \neq 0$.

Consider row i^* of D, and consider its product with vector $x \in \{0, 1\}^n$:

$$\sum_{j=1}^{n} D_{i^* j} \cdot x_j.$$

This gives the value for *position* i^* in the length-n vector computed by $D \cdot x$.

We will show that this value will be 0 with probability at most $1/2$.

Analysing MMVerify: $AB \neq C$

When drawing a random $x \in \{0,1\}^n$ uniformly at random (u.a.r.), each x has equal probability $(1/2^n)$.

This is equivalent to choosing the values $x_i \in \{0,1\}$ independently with probability $1/2$, for each $i \in [n] = \{1, \ldots, n\}$.

Use this in the analysis (*principle of deferred decisions*).
Write $\sum_{j=1}^{n} D_{i^*j} \cdot x_j$ as

$$\left(\sum_{j \in [n] \setminus \{j^*\}} D_{i^*j} \cdot x_j \right) + D_{i^*j^*} \cdot x_{j^*}$$

Think about sampling x (*deferred decisions*) as a $\{0,1\}^{n-1}$ vector first, followed by the value for x_{j^*} last.

Analysing MMVerify: $AB \neq C$

After sampling the $\{0,1\}^{n-1}$ vector for positions $\{x_j \mid j \in [n] \setminus j^*\}$, we now have a fixed value for

$$\sum_{j \in [n] \setminus \{j^*\}} D_{i^* j} \cdot x_j.$$

Then no matter over which "ring" our arithmetic is in (whether integers, or rationals, or even a finite field), there is *at most one* value which could be added to this to get 0 (maybe 0, maybe 1, maybe some other non-zero value).

Also, we know $D_{i^* j^*} \neq 0$. Sampling x_{j^*} last, we will get $D_{i^* j^*} \cdot x_{j^*} = D_{i^* j^*}$ (which is non-zero) with prob. $1/2$, and $D_{i^* j^*} \cdot x_{j^*} = 0$ with prob. 1/2. Hence

$$\Pr\left[\sum_{j=1}^{n} D_{i^* j} \cdot x_j = 0\right] \leq 1/2$$

All trials of MMVerify: $AB \neq C$

Previous slides present the analysis of what happens ($AB \neq C$ case) on a single sample from $\{0,1\}^n$ (tested in lines 2.-7. of Algorithm MMVERIFY).

- The Algorithm is set up to **return** “no” (and terminate) on the first trial where it discovers a mismatch between $AB \cdot x$ and $C \cdot x$.
- It only **return**s “yes” if it passed through all k iterations of the loop with all trials giving a match.
- “Every trial gives a match” is the bad event for analysing the $AB \neq C$ case.

All trials of MMVerify: $AB \neq C$

Notice that the k repeated trials fit into the paradigm of "sampling with replacement".

Let E_j be the event that the j-th sampled x satisfies $D \cdot x = 0$ (i.e., $AB \cdot x = C \cdot x$).

$E_1, \ldots, E_k$ are all mutually independent. Thus, applying Defn 1.3 from lecture 1,

$$\Pr[\cap_{j=1}^{k} E_j] = \prod_{j=1}^{k} \Pr[E_j].$$

We have already shown that $\Pr[E_j] \leq 1/2$.

Hence $\Pr[\cap_{j=1}^{k} E_j]$, the probability that the algorithm returns "YES" is at most $1/2^k$ (in the case of $AB \neq C$).

This completes the proof that with k repeated trials the probability of error (an incorrect answer) by the algorithm is at most $1/2^k$. $\Box$

Reading Assignment

Continue reading Chapter 1 of "Probability and Computing".