Randomness and Computation
or, “Randomized Algorithms”

Mary Cryan
School of Informatics
University of Edinburgh

Testing polynomial identities

We are given two polynomials $F(x)$ and $G(x)$, where $F(x)$ is expressed as a product of $d$ "monomials" and $G(x)$ is given as an expansion of $x^i$ terms, with degree at most $d$.

How much time does it take to verify whether $F(x) \equiv G(x)$?

Verifying polynomial identities using random sampling

From Tuesday (17th), we have the following randomised algorithm:

- Choose a value for $x$ uniformly at random from the set of integers $\{1, \ldots, 100d\}$.
- Calculate $F(x)$ for this value, taking $\Theta(d)$ time.
- Calculate $G(x)$ for this value, taking $\Theta(d)$ time.
- Compare the two numbers ... answering “yes” if they are the same, “no” otherwise.

Recall that uniformly at random means “all values of the set have the same chance of being generated” (in his case, that chance is $1/100d$).

We showed:

- If $F(x)$ really is the same polynomial as $G(x)$, the random process always says “yes”.
- If $F(x)$ is different from $G(x)$, then it says “no” with probability greater than or equal to $99/100$.

Probability

When dealing with random variables and their outcomes, we use the concepts of probability and expectation.

Definition (1.1)

A probability space has three components:

1. A sample space $\Omega$, the set of all possible outcomes of the random process modelled by the probability space;
2. A family of sets $\mathcal{F}$ representing the allowable events, each set in $\mathcal{F}$ being a subset of the sample space $\Omega$; and
3. A probability function $\Pr : \mathcal{F} \to \mathbb{R}^+$, satisfying the laws of probability in the next definition.
Probability cont'd.

Definition (1.2)
A probability function is any function \( f : \mathcal{F} \to \mathbb{R}^+ \) satisfying the following conditions:
1. For any event \( F \in \mathcal{F} \), \( 0 \leq \Pr[F] \leq 1 \);
2. \( \Pr[\Omega] = 1 \);
3. For any finite or countably infinite sequence of pairwise mutually disjoint events \( F_1, F_2, \ldots \),
\[
\Pr \left( \bigcup_{i \geq 1} F_i \right) = \sum_{i \geq 1} \Pr[F_i].
\]

In RC, we usually work with a discrete probability (\( \Omega \) being finite or countably infinite) with \( \mathcal{F} \) containing all subsets of \( \Omega \).

Refining the verification of polynomial identities

Imagine we now run two random trials to test \( F(x) \equiv G(x) \), first drawing a random \( x_1 \) from \( \{1, \ldots, 100d\} \) and testing whether \( F(x_1) \equiv G(x_1) \), next drawing a random \( x_2 \) from \( \{1, \ldots, 100d\} \) and testing whether \( F(x_2) \equiv G(x_2) \).
We return "yes" if both calculations give matching values, otherwise we return "no".

Observation
Notice that this refined algorithm again gives one-sided error:
- If \( F(x) \) and \( G(x) \) are the same polynomial, certainly we will see that \( F(x_1) \) matches \( G(x_1) \), and that \( F(x_2) \) matches \( G(x_2) \) (answer "yes").
- If \( F(x) \) and \( G(x) \) are non-identical, then the algorithm returns "no" most of the time, with failure probability at most \( 1/(100)^2 \).

Refining the verification of polynomial identities (analysis)

We have two options for "repeated sampling" from \( \{1, \ldots, 100d\} \), with replacement or without replacement.

With replacement: We draw the random value \( x_2 \) uniformly at random from \( \{1, \ldots, 100d\} \) (including \( x_1 \) as an option).
For this case, the two events of "generating \( x_1 \)" and "generating \( x_2 \)" are mutually independent.

Definition (1.3)
The two events \( F \) and \( G \) are said to be mutually independent if and only if
\[
\Pr[F \cap G] = \Pr[F] \cdot \Pr[G].
\]
Refining the verification of polynomial identities (analysis)

with replacement (cont'd): Recall that if \( F(x) \neq G(x) \), then \( F(x) - G(x) \) has at most \( d \) roots; hence there are at most \( d \) values in \( \{1, \ldots, 100d\} \) that could give matching values for \( F(x), G(x) \).

If \( H_1 \) is the event that "a root of \( F(x) - G(x) \)" is generated on this first trial, then \( \Pr[H_1] \leq d/100d = (1/100) \).

But sampling with replacement, the outcomes of the second trial, are independent of what happened before. So \( H_2 \) (the probability of generating a root of \( F(x) - G(x) \) on the second trial) is independent of \( H_1 \) (with identical probability).

The probability that both experiments would draw a root of \( F(x) - G(x) \) is (by Defn 1.3) equal to

\[
\Pr[H_1] \cdot \Pr[H_2] \leq (1/100) \cdot (1/100),
\]

which is at most \( 1/100^2 \).

Refining the verification of polynomial identities (analysis)

without replacement: We have already tested \( x_1 \) and found \( F(x_1), G(x_1) \) to match, for \( H_2 \) we will draw a value from \( \{1, \ldots, 100d\} \setminus \{x_1\} \).

Events \( H_1 \) and \( H_2 \) are no longer independent, \( H_2 \) is conditional on \( H_1 \).

Definition (1.4)

The conditional probability of event \( E \) conditional on event \( F \) having happened is

\[
\Pr(E \mid F) = \frac{\Pr[E \cap F]}{\Pr[F]}.
\]

Refining the verification of polynomial identities (analysis)

without replacement cont'd: In applying Definition 1.4, \( E \) is \( H_1 \) and \( F \) is \( H_2 \). We want to calculate \( \Pr[H_1 \cap H_2] \) (two samples both giving a false match). This is \( \Pr[H_1] \cdot \Pr[H_2 \mid H_1] \).

We know \( \Pr[H_1] = \frac{d'}{100d} \) for some value \( d' \leq d \) (number of different roots of \( F(x) - G(x) \) in \( \{1, \ldots, 100d\} \)).

Also we know \( d' \geq 1 \) (we only bother to sample \( x_2 \) if \( F(x_1), G(x_1) \) were a match).

For \( \Pr[H_2 \mid H_1] \), note that since \( H_1 \) occurred, we only have \( d' - 1 \) roots of \( F(x) - G(x) \) in the set \( \{1, \ldots, 100d\} \setminus \{x_1\} \).

Hence \( \Pr[H_2 \mid H_1] = \frac{d' - 1}{100d} \). hence ....

\[
\Pr[H_1 \cap H_2] = \frac{d'}{100d} \cdot \frac{d' - 1}{100d - 1} = \frac{d'(d' - 1)}{100d(100d - 1)} < \frac{1}{100^2}.
\]

Can similarly consider carrying out \( k \) different trials of values sampled from \( \{1, \ldots, 100d\} \).

- Will be able to show “one-sided error” of at most \( 1/100^k \).
- The probability of failure (returning “yes” when \( F(x), G(x) \) are non-identical) is always a bit better in the “without replacement” case).
- This iterated testing algorithm will take \( \Theta(k \cdot d) \) time.
- No point doing more than \( d \) iterations (why?)
Verifying Matrix Multiplication

First topic for next Tuesday’s lecture will be on verification of Matrix Multiplication.

We are given three $\times n$ matrices $A, B, C$.
How efficiently can we verify whether $AB = C$ (without multiplying $A$ by $B$)?

Little bit harder than the polynomial testing question (but more interesting)!!

Note that “multiply-out” for Matrix Multiplication takes $\Theta(n^3)$ via the “high school” method. The “quickest” deterministic algorithm for multiplying out has complexity around $\Theta(n^{2.37})$.

Reading Assignment

Continue reading Chapter 1 of “Probability and Computing”.