Pseudorandomness

Boran Erol

March 2024

1 Introduction and Motivation

The primer to the theory of pseudorandomness is based on a fresh view at the *question of randomness*, which has been taken by complexity theory. The idea is to call everything random if we can't efficiently distinguish it from the uniform random distribution. Therefore, randomness is not an inherent property of the object, it's dependent on the observer.

At the extreme, this approach says that the question of whether the world is determinstic or allows for some free choice (randomness) is irrelevant. All it matters is that how the world looks to us and computationally bounded devices.

Three fundamental aspects of PRGs:

1. The stretch
2. The algorithms it fools (computational indistinguishability)
3. The computational complexity of the PRG algorithm

Three theories of randomness:

1. Shannon's Information Theory: Randomness is lack of information.
2. Kolmogorov Complexity: Randomness is lack of structure.
3. Complexity Theoretic View

The generation process is fixed before the distinguisher. Therefore, even though the PRG runs in fixed polynomial time, it's indistinguishable to any PPT distinguisher. In particular, the distinguisher is allocated more resources than the generator. Moreover, the distinguisher is probabilistic but the generator is deterministic.

However, in the case of derandomization, it's more natural to do the opposite. Instead, we fix the observer first and base the generator on it.

Definition 1. A deterministic polynomial time algorithm G is a PRG if there's a stretch function $l : \mathbb{N} \to \mathbb{N}$ (satisfying $\forall k : l(k) > k$) such that for all PPT's D, for all positive polynomials p, and for all sufficiently large k we have that

$$|\Pr[D(G(U_k)) = 1] - \Pr[D(U_{l(k)}) = 1]| \leq \frac{1}{p(k)}$$

where the probability is taken over U_k and the coin tosses of D.

PRGs can be used to reduce the randomness complexity of randomized algorithms without making a noticeable difference in their output.

Statistically close distributions are trivially computationally indistinguishable.

You can upgrade single-sample indistinguishability to multi(polynomial)-sample indistinguishability by using a hybrid argument where the neighboring distributions differ by a single sample.

2 Pseudorandomness Workshop at Simon's Institute

- Make the probabilistic method constructive. Construct *efficiently* and deterministically objects whose existence is guaranteed by proofs based on the probabilistic method. Efficiency is crucial since otherwise we get constructive proofs by enumeration.
- Convert a randomized algorithm for a problem of interest into a deterministic algorithm of *comparable complexity*.
- *Efficiently* construct deterministically (or with little true randomness) objects having many of the useful properties of random objects.

Theorem 2.1. Every n-vertex graph has either a clique or an independent set of size at least $\frac{1}{2}\log n$.

Theorem 2.2. There's an n-vertex graph in which the max clique and the maximum independent set has size at most $2\log n$.

80-year-old open problem: Match Erdos's existence proof with an explicit construction.

Shannon's second theorem: If any encoding scheme works, a random encoding scheme works with high probability.

The issue is that encoding schemes have doubly exponential sizes and it takes doubly exponential time to decode. Lots of work in the last 70 years to make these constructions explicit.

Derandomization of Algorithms

1. Primality Testing
2. Finding Large Primes
3. Polynomial Identity Testing
4. Approximating the permanent
5. Circuit acceptance

If the ideal PRG exists, $BPP = P$, all proofs using the probabilistic method can be efficiently constructed and all randomized algorithms can be randomized.

Notice that we can fool one-bit tests by just repeating a single bit. This has a cool story behind it that's told in Minute 30 of the Fundamental Techniques of Pseudorandomness II talk by Trevisan.

3 k-wise Independence

Let $z \in F_2^n$. Let $\phi_z : F_2^n \to F_2$ be the map defined by $\phi_a = \langle z, a \rangle$. Then, the nullity of this map is $n-1$ and therefore the probability that a randomly sampled a is in the kernel of this map is $1/2$.

Even simpler, this is a linear equation in F_2 so the space of solutions is $n-1$ dimensional.

Now notice that if you have $a \neq a'$ and consider $\langle z, a \rangle = 0, \langle z, a' \rangle = 0$, you get two linearly independent equations in F_2, so the space of solutions has dimension $n-2$ no matter whether the right-hand side is $(0,0), (0,1), (1,0), (1,1)$.

In fact, this proves that linear independence translates into stochastic independence in this setting!

Pairwise independence can also be seen as a PRG that fools tests that only look at two bits.

Let's now generalize this idea. Our aim is to do the following: given t random bits, we want to construct a function h :

For every $x \neq y$, over the randomness of a, b the two values $ax+b$ and $ay+b$ are uniform and independent.

This is a little confusing: we'll think of x, y as constants and a, b as variables.

Notice that this means there are $2^n \times 2^n = 2^{n+1}$ possible values for this tuple and every value has the same probability, namely,

$$\frac{1}{2^n + 1}$$

This follows from the fact that the linear system of equations given by

$ax + b = v$ and $ay + b = w$

has a unique solution since the two left hand-sides are linearly independent.

Thus, we can generate a function $h : \{0,1\}^n \to \{0,1\}^n$ using $2n$ bits. This is not the most optimal, we can get it down to n bits.

Notice that if $m < n$, we can use this construction and just restrict the function to satisfy the condition.

If $m > n$, we can construct a function from m bitstrings to other m bitstrings and pad all strings with 0s. Therefore, we only use $2 \times \max m, n$ bits.

Notice that setting $m = 1$ gives a worse pairwise generator than the Hadamard code. In fact, $\max(m, n)$ is an optimal lower bound for t.

4 Pseudorandomness and Regularity in Graphs

Three different regimes:

- The sparse case - d = $O(1)$.
- The dense case - d = $\Omega(n)$.
- The intermediate case.

Random graphs and functions are expanders and extractors with high probability. These are also useful sources for constructing PRGs.

5 Expanders

An expander can informally be defined as a graph where every set of nodes has many neighbors.

For example, we want networks to be expander graphs since non-expanding subsets correspond to bottlenecks.

The definition of expanders is combinatorial. There are two different perspectives that are useful: probabilistic and algebraic.

- Combinatorial: Definition
- Probabilistic: Random walks on rapidly mixing graphs
- Algebraic: Second largest eigenvalue λ_2

Let $G = (V, E)$ be a d-regular graph with n vertices.

We can talk about vertex expansions and edge expansions.

Notice that $h_v \leq h_e \leq d \times h_v$. Since we usually consider constant degree graphs, we don't care about the distinction.

Balancing two contradicting properties: the graph should be sparse and connected at the same time.

Another way to intuit this is as a graph that is hard to cut.

This reminded of the minimum cut and maximum flow.

6 Extractors

Our goal is to get some weak source of randomness (biased randomness) and try to bits that look randomness.

Creating pure randomness from a single weak source of randomness is impossible. Thus, we explore 3 different relaxations:

- Seeded Extractors
- Multi-source extractors
- Seedless extractors for structured sources