COMS E6998: Topics in Advanced Complexity Theory Spring 2024

Unconditional Lower Bounds and Derandomisation

Prof. Rocco Servedio *Scribe: Ekene Ezeunala*

Contents

Conventions for notes.

1. $\mathbb{N} = \{1, 2, 3, \ldots\}$, $\mathbb{N}_0 = \{0, 1, 2, 3, \ldots\}$, $[n] = \{1, 2, \ldots, n\}$, $[n]_0 = \{0, 1, \ldots, n\}$.
2. $\lg n = \log_2 n$.
3. $a \approx_\varepsilon b$ means $|a - b| \leqslant \varepsilon$.
4. $\log^* n$ is the number of times you have to apply log to n before you get a number $\leqslant 1$.
5. $\mathcal{U}_s$ is the uniform distribution over $\{0,1\}^s$ or $\{\pm 1\}^s$. $\mathcal{N}(0,1)$ is the standard normal distribution, and $\mathcal{N}(\mu, \sigma^2)$ is the normal distribution with mean μ and variance σ^2.
6. $\mathbb{E}[f]$ means $\mathbb{E}_{\boldsymbol{u} \sim \mathcal{U}}[f(\boldsymbol{u})]$ and $\Pr[f]$ means $\Pr_{\boldsymbol{u} \sim \mathcal{U}}[f(\boldsymbol{u})]$.
7. We will use bolded font for random variables, e.g. $\boldsymbol{u}$, and calligraphic font for distributions, e.g. $\mathcal{D}$. The only exception is when we use $\mathcal{C}$ to denote a class of functions.

§1 Lecture 01—16th January, 2024

Overview of today's lecture.

- Introductions: course, instructor, students.
- Administrative and logistical overview.
- More detailed overview of course topics.

§1.1 Introduction

For this course we will be using the excellent survey by Hatami and Hoza [HH23].

The general topic for this course is complexity theory. There are two general strands of complexity theory:

1. *High-level complexity:* This strand dwells with angels in the Great Beyond and deals with "high level" complexity questions: Does randomness enhance the power of efficient computation (i.e. is P properly contained in BPP)? Is it easier to verify a proof than to construct one (i.e. is P properly contained in NP)? Of course, we do not know the answers to either of these questions; most results in "high level" complexity are conditional results that rely on various unproven assumptions such as $\mathsf{P} \neq \mathsf{NP}$ or $\mathsf{P} = \mathsf{BPP}$. While many interesting connections have been established between different computational problems and computational resources, and many beautiful and important results have been achieved, the major open questions here are unanswered and seem likely to stay that way for the foreseeable future.
2. *Low-level complexity:* A second strand of research in complexity theory is rooted down in the mud with the worms and grubs; it deals with establishing unconditional results. This is essentially a "low level" study of computation; it typically centers around simple models of computation such as decision trees and branching programs, Boolean formulas and restricted classes of Boolean circuits, different types of polynomials, and so on. In this line of research unconditional results are established which rely on no unproven assumptions, but we are only able to achieve such results for various restricted models of computation such as the examples given above. There has been steady progress made here over the years using a range of techniques from combinatorics, algebra, analysis, probability, and other branches of mathematics. Results and techniques from this low-level study of computation often play a role in "high-level" complexity.

In this class we will look at some of the following:

Restricted models of computation	Different research goals
Boolean formulas	Worst-case lower bounds
Shallow Boolean circuits	Average-case lower bounds
DNF and CNF formulas	Pseudorandom generators (PRGs)
Decision trees	Pseudorandom functions (PRFs)
Branching programs	Pseudorandom permutations (PRPs)
Polynomial models	Pseudorandom generators for low-degree polynomials
Communication protocols	Deterministic approximate counting algorithms

For the entire class we will be establishing a connection between the two columns of the table above; for example we may be concerned with proving average-case lower bounds for shallow Boolean circuits or with constructing pseudorandom generators for certain polynomial models of computation. We will not cover all of the topics in the table above, but we will cover a good number of them.

§1.2 Overview of course topics

First a few definitions:

Definition 1.1 (Boolean function)**.** *A* Boolean function $f(x) = f(x_1, \ldots, x_n)$ *is a function from* $\{0,1\}^n$ *to* $\{0,1\}$.

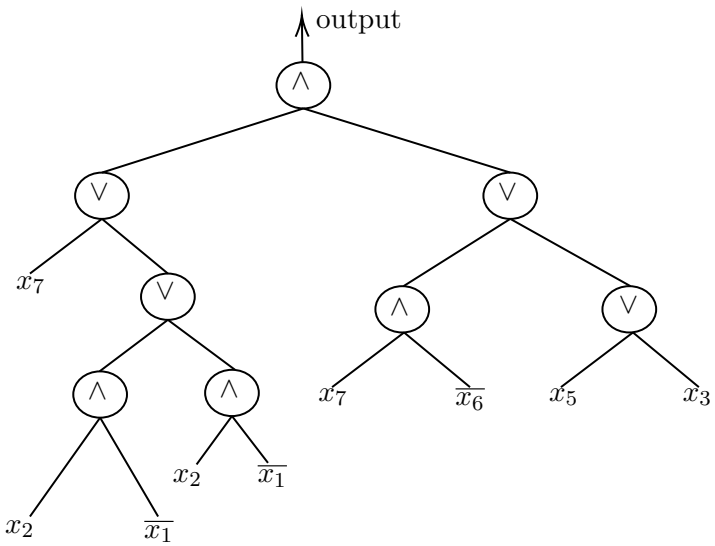
Let $\mathcal{B}$ be a set of Boolean functions, which we often call a basis set. The fan-in of a function $g \in \mathcal{B}$ is the maximum number of inputs to any gate in the circuit computing g.

Definition 1.2 (Boolean circuit)**.** *A* Boolean circuit *over a basis set* $\mathcal{B}$ *is a directed acyclic graph (DAG) such that for each node* v *in the graph, either (a)* v *has indegree* 0 *and is labelled with an input variable* x_i *for some* $i \in \mathbb{N}$*, or (b)* v *has indegree* $n > 0$ *and is labelled with a function* $g \in \mathcal{B}$*, where* g *is an* n*-ary Boolean function and with an ordering on the incoming edges to* v*. The output of the circuit is the label of the node with indegree* 0.

A gate is a non-input vertex in a Boolean circuit, the size of a circuit is the number of gates in the circuit, the depth of a circuit is the maximum number of gates on any path from an input to the output, and the width of a circuit is the maximum number of gates at any level of the circuit.

Here are some computational models that we will be looking at:

1. **Boolean formulas.** A Boolean formula is a Boolean circuit in which every gate has outdegree less than or equal to 1. Here's an example of a Boolean formula:



   We will discuss Boolean formulas in greater detail throughout the course. Indeed, we will see that Boolean formulas are a very useful model of computation for proving lower bounds on the size of Boolean circuits.

Example 1.3. Define the class of Boolean functions

$$\begin{aligned}\mathrm{FORM}_S &= \mathrm{FORM}_{S(n)} \\ &= \{\text{class of } f : \{0,1\}^n \to \{0,1\} \text{ comput. by a Boolean formula of size} \leqslant S\}.\end{aligned}$$

In the next lecture we will show that $\mathsf{PARITY}_n \notin \mathrm{FORM}_{n^{1.49}}$—this is an example of a *worst-case* lower bound.

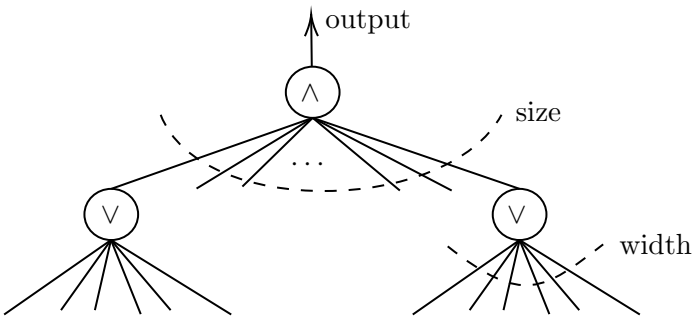
2. **DNFs and CNFs.** A *disjunctive normal form* (DNF) is a Boolean formula of the form

$$f(x_1, \ldots, x_n) = \bigvee_{i=1}^{m} \bigwedge_{j=1}^{n} \ell_{ij},$$

where each ℓ_{ij} is either x_k or $\overline{x_k}$ for some $k \in [n]$. A *conjunctive normal form* (CNF) is a Boolean formula of the form

$$f(x_1, \ldots, x_n) = \bigwedge_{i=1}^{m} \bigvee_{j=1}^{n} \ell_{ij},$$

where each ℓ_{ij} is either x_k or $\overline{x_k}$ for some $k \in [n]$. We will discuss DNFs and CNFs in greater detail throughout the course. Here's an example of a DNF:



We can similarly define a size-s-DNF as a DNF with s clauses and a width-w-DNF as a DNF with w variables in each clause. We can similarly define size-s-CNFs and width-w-CNFs. We will see that DNFs and CNFs are useful models of computation for proving lower bounds on the size of Boolean circuits.

3. $\mathrm{AC}^0_{d,s}$**.** The class $\mathrm{AC}^0_{d,s}$ is the class of Boolean functions computable by a circuit of depth d, polynomial size, and unbounded fan-in. We will discuss $\mathrm{AC}^0_{d,s}$ in greater detail throughout the course.

4. $\mathbb{F}_2$**-polynomials.** The class of $\mathbb{F}_2$-polynomials is the class of polynomials over the finite field with two elements, $\mathrm{GF}(2) = \mathbb{F}_2 = \{0,1\}$ in n variables. An $\mathbb{F}_2$-polynomial is a polynomial of the form

$$f(x_1, \ldots, x_n) = \sum_{S \subseteq [n]} a_S \prod_{i \in S} x_i,$$

where $a_S \in \mathbb{F}_2$ for each $S \subseteq [n]$. For polynomials $f, g \in \mathbb{F}_2[x_1, \ldots, x_n]$, we say that f and g are equal if and only if their coefficients are equal. We will often discuss $\mathbb{F}_2$-polynomials in terms of their sparsity—the number of contributing monomials in the polynomial—and their degree—the maximum degree of any monomial in the polynomial. We will discuss $\mathbb{F}_2$-polynomials in greater detail throughout the course.

Example 1.4. We can define the following class of $\mathbb{F}_2$-polynomials:

$$\begin{aligned}\mathrm{DEG}_d &= \mathrm{DEG}_{d(n)} \\ &= \{\text{all } f : \{0,1\}^n \to \{0,1\} \text{ computable by a degree-}d\ \mathbb{F}_2\text{-polynomial}\}.\end{aligned}$$

Note that here n is an asymptotic parameter.

§1.3 Goals for our area of study

We have fleshed out the left side of the table from earlier, but what about the right side? What are the goals of our area of study?

§1.3.1 Establishing hardness

How do we establish hardness? The dream is to establish a proof that $\mathsf{P} \neq \mathsf{NP}$:

Problem 1.5 ($\mathsf{P} \neq \mathsf{NP}$)**.** *Is every language L accepted by some nondeterministic algorithm in polynomial time also accepted by some deterministic algorithm in polynomial time?*

We think that there are natural Boolean functions which don't have poly-time algorithms or poly-sized formulas to compute them or poly-sized $\mathbb{F}_2$-polynomials to compute them. We think this is true (at least most people do), but there has been very little progress towards proving this result. But how can we hope to prove this? It is known that $\mathsf{P} \neq \mathsf{NP}$ if we have worst-case superpolynomial lower bounds on Boolean circuits computing functions which correspond to languages $L \in \mathsf{NP}$. In particular,

> If it was true that any family of unrestricted Boolean circuits $(C_n)_{n \geqslant 1}$ of circuits solving the Hamiltonian cycle problem Ham-Cycle (or any other NP-complete problem for that matter) on n-node graphs must have size $\geqslant n^{\omega(1)}$ (i.e. superpolynomial in n), then it would follow as a consequence that $\mathsf{P} \neq \mathsf{NP}$.

This establishes strong circuit-lower bounds as a natural goal for our area of study. But we don't know how to prove this; in fact we are very far from proving this. So we need to lower our sights and look at weaker goals instead. We can look at proving lower bounds for restricted models of computation like Boolean formulas, DNFs, CNFs, $\mathrm{AC}^0_{d,s}$ circuits, polynomial threshold functions, $\mathbb{F}_2$-polynomials, etc. The following definition is natural given the above discussion:

Definition 1.6 (Worst-case hardness)**.** *Let $\mathcal{C}$ be a class of functions $f : \{0,1\}^n \to \{0,1\}$ (for example $\mathcal{C} = \mathrm{FORM}_{n^{1.49}}$). We say that $h : \{0,1\}^n \to \{0,1\}^n$ is* worst-case hard for $\mathcal{C}$ *if $h \notin \mathcal{C}$.*

The way to think of this is that for every function in the class $\mathcal{C}$, for every Boolean formula, however clever, there is some input on which the formula fails to compute the function correctly—the worst enemy gets to adversarially pick such an input. A stronger lower-bound notion is that of an *average-case hard* function; instead of worrying about whether there's a function in $\mathcal{C}$ which computes h correctly on *every* input, perhaps we can consider instead a function in $\mathcal{C}$ which gets h

right on almost every input—hence average-case hardness. Before we define this, we first need to set things up:

Definition 1.7 (Correlation)**.** *Let $f, g : \{0,1\}^n \to \{0,1\}$ be two Boolean functions, and let $\mathcal{D}$ be some distribution over $\{0,1\}^n$. Define the* correlation Cor *as a measure of the failure probability of f while computing g over $\mathcal{D}$:*

$$\mathrm{Cor}_{\mathcal{D}}[f,g] \coloneqq \left| \Pr_{\boldsymbol{x}\sim\mathcal{D}}[f(\boldsymbol{x}) = g(\boldsymbol{x})] - \Pr_{\boldsymbol{x}\sim\mathcal{D}}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})] \right| = 2 \cdot \left| \frac{1}{2} - \Pr_{\boldsymbol{x}\sim\mathcal{D}}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})] \right| \in [0,1].$$

The correlation is simply a measure of how often f and g agree on inputs drawn from $\mathcal{D}$. If they agree (or disagree) with each other half the time, then the correlation is 0; if they agree (or disagree) with each other all the time, then the correlation is 1. We can also speak of the correlation of a fixed Boolean function $g : \{0,1\}^n \to \{0,1\}$ with a class of Boolean functions $\mathcal{C}$:

$$\mathrm{Cor}_{\mathcal{D}}[g,\mathcal{C}] \coloneqq \max_{f\in\mathcal{C}} \mathrm{Cor}_{\mathcal{D}}[f,g].$$

Often $\mathcal{D}$ is the uniform distribution $\mathcal{U} = \mathcal{U}_n$ over $\{0,1\}^n$, in which case we write $\mathrm{Cor}_n[f,g]$ and $\mathrm{Cor}_n[g,\mathcal{C}]$ (or even omit the subscript n altogether).

Remark 1.8. *1. If $f \equiv g$ or $f \equiv \overline{g}$, then $\mathrm{Cor}_{\mathcal{D}}[f,g] = 1$ (obviously).*

2. If $f, g : \{0,1\}^n \to \{\pm 1\}$, then

$$\mathrm{Cor}_{\mathcal{D}}[f,g] = \left| \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[f(\boldsymbol{x}) \cdot g(\boldsymbol{x})\right] \right|.$$

We can now define average-case hardness:

Definition 1.9 (Average-case hardness)**.** *Let $\mathcal{C}$ be a class of functions $f : \{0,1\}^n \to \{0,1\}$, $\mathcal{D}$ be a distribution over $\{0,1\}^n$, and take a parameter $\varepsilon > 0$. We say that $h : \{0,1\}^n \to \{0,1\}$ is* ε-hard for $\mathcal{C}$ under $\mathcal{D}$ *if for all functions $f \in \mathcal{C}$, we have that*

$$\left| \Pr_{\boldsymbol{x}\sim\mathcal{D}}[f(\boldsymbol{x}) = h(\boldsymbol{x})] - \frac{1}{2} \right| \leqslant \varepsilon,$$

that is, $\mathrm{Cor}_{\mathcal{D}}[f,\mathcal{C}] \leqslant 2\varepsilon$.

Think of ε in the definition above as the advantage over triviality—for smaller ε, the function h is harder to compute than for larger ε. Thus proving average-case hardness for a function h under a distribution $\mathcal{D}$ is a stronger notion than proving worst-case hardness for h.

§1.3.2 Getting rid of randomness

In computation, the use of probabilistic tools and randomness has often led to the development of new algorithms and proof techniques. Randomised algorithms—algorithms that toss coins and make random decisions—have been used to develop algorithms for a variety of problems, and these algorithms are often simpler and more efficient than their deterministic counterparts. Here are two examples:

Problem 1.10 (Graph reachability)**.** *Given an n-node undirected graph $G = (N, E)$ and two vertices $s, t \in N$, does there exist a path from s to t in G?*

This problem has many deterministic algorithms; a popular example is the breadth-first search algorithm. However, there is a very simple randomised algorithm for this problem: simply perform a random walk on the graph starting from s and take $100n^2$ steps; if t is reachable from s, then the probability that we reach t is $\Pr[\text{hit } t] \geqslant 0.99$. This algorithm requires space $O(\log n)$ and time $O(n^2)$, and it is very simple to implement.

Problem 1.11 (Polynomial identity testing)**.** *Given two algebraic formulas $p(x_1, \ldots, x_n)$ and $q(x_1, \ldots, x_n)$, consisting only of the operations $+, \times$, are p and q equal as polynomials for all inputs $x_1, \ldots, x_n$?*

The only known algorithm for this problem is randomised: define the quantity $m = |p| + |q|$, and pick $\boldsymbol{x} = (x_1, \ldots, x_n)$ uniformly from $\{1, 2, \ldots, 3m\}^n$, and then check if $p(x) = q(x)$. It can be shown that if $p \not\equiv q$, then

$$\Pr[p(\boldsymbol{x}) = q(\boldsymbol{x})] \leqslant \frac{\deg(p - q)}{3m} \leqslant \frac{1}{3},$$

and we can repeat this procedure sufficiently many times to reduce the probability of error to a very small constant.

Randomness is crucial in both of these algorithms, and many more! But in some cases we may want to devise algorithms and heuristics that guarantee correct answers with absolute certainty, and, in particular, determine whether fast randomised algorithms have fast deterministic counterparts. This is at the heart of the following problem:

Problem 1.12 ($\mathsf{BPP} = \mathsf{P}$)**.** *Does every polynomial time randomised algorithm have an efficient deterministic analogue?*

We believe that the answer to this question is yes, but we don't know how to prove this. Indeed, there is a long history of randomised algorithms successfully being derandomised: two examples are the derandomisation of the Graph isomorphism problem accomplished by Reingold, Trevisan, and Vadhan 25 years after the original randomised algorithm was proposed [RTV06], and the derandomisation of the Primality testing problem accomplished by Agrawal, Kayal, and Saxena 30 years after the original randomised algorithm was proposed [AKS04].

Notwithstanding the above, how might we go about proving $\mathsf{BPP} = \mathsf{P}$?

> If there exists a "good-enough" explicit pseudorandom generator (PRG) $G : \{0,1\}^n \to \{0,1\}^m$ (i.e. a function G such that $G(U_n)$ "looks like" U_m) for the class $\mathcal{C}$ of all polynomial-sized Boolean circuits, then it would follow as a consequence that $\mathsf{BPP} = \mathsf{P}$.

This establishes the appeal of a good-enough explicit pseudorandom generator as a natural goal for our area of study. We now define pseudorandom generators:

Definition 1.13 (Fooling)**.** *Let $f : \{0,1\}^n \to \{0,1\}$, and let $\boldsymbol{X}$ be a random variable over $\{0,1\}^n$. We say that $\boldsymbol{X}$ ε-fools f if*

$$|\mathbb{E}[f(\boldsymbol{X})] - \mathbb{E}[f(\mathcal{U}_n)]| \leqslant \varepsilon,$$

where $\mathcal{U}_n$ is the uniform distribution over $\{0,1\}^n$.
We say that X ε-fools a class $\mathcal{C}$ of functions *if X ε-fools every function $f \in \mathcal{C}$.*

Definition 1.14 (Pseudorandom generators (PRGs) for a class of functions)**.** *Let $f : \{0,1\}^n \to \mathbb{R}$, $G : \{0,1\}^s \to \{0,1\}^n$, and $\varepsilon > 0$. Then we say that G is an* ε-PRG for f *(i.e. G fools f with error ε) if $G(U_s)$ ε-fools f, where U_s is the uniform distribution over $\{0,1\}^s$.*
Similarly, G is an ε-PRG for a class $\mathcal{C}$ of functions *(i.e. G fools $\mathcal{C}$ with error ε) if G is an ε-PRG for every function $f \in \mathcal{C}$.*

The idea here is that given a "seed-length" $s \ll n$, we can toss s coins (hence $\mathcal{U}_s$) and not have to toss n coins (hence $\mathcal{U}_n$) to get a function that "looks like" f and is good enough for our purposes. Equivalently, a PRG G uses a few truly random bits (the s coin tosses) to sample a pseudorandom string that is indistinguishable from a truly random string, from the perspective of the observer f.

Despite the promising claims of PRGs, there is one unfortunate fact: for any non-trivial PRG, there exists a function that is not folled by it.

Theorem 1.1 (Impossibility of fooling all functions)**.** *Let $G : \{0,1\}^s \to \{0,1\}^n$ where $s \ll n$. Then there exists a function $f : \{0,1\}^n \to \{0,1\}$ such that G does not 0.49-fool f.*

Proof. Let f be a Boolean function outputting 1 in the image of G. When we draw $\boldsymbol{x} \sim \mathcal{U}_n$, the probability that it remains in the image of G is $2^s/2^n$ (since there are 2^s strings in the image of G and 2^n strings in $\{0,1\}^n$). Therefore, for $\varepsilon = 0.49$,

$$\left| \Pr_{\boldsymbol{x} \sim G(\mathcal{U}_n)} [f(\boldsymbol{x}) = 1] - \Pr_{\boldsymbol{x} \sim \mathcal{U}_n} [f(\boldsymbol{x}) = 1] \right| = \left| 1 - \frac{2^s}{2^n} \right| > 0.49.$$

Thus G does not 0.49-fool f. □

We say that a PRG G is *explicit* if it is computable in polynomial time. Indeed, nonexplicit PRGs exist:

Lemma 1.15 (Nonexplicit PRGs exist and are easy)**.** *Let $\mathcal{C}$ be any class of Boolean functions $f : \{0,1\}^n \to \{0,1\}$, and take $\varepsilon > 0$. Then there exists an ε-PRG for $\mathcal{C}$ with seed-length*

$$\log \log |\mathcal{C}| + 2 \log \frac{1}{\varepsilon} + O(1).$$

Proof. Let $G : \{0,1\}^s \to \{0,1\}^n$ be uniformly random; $G(y)$ is uniformly random in $\{0,1\}^n$ for every $y \in \{0,1\}^s$. For each fixed Boolean function f and fixed s-bit y, $f(G(y))$ is a random bit in $\{0,1\}$ with $\mathbb{E}_G [f(G(y))] = \mathbb{E}_{\mathcal{U}_n} [f(\mathcal{U}_n)]$. We have independence, and therefore by the Chernoff bound,

$$\Pr_G \left[\left| \mathbb{E}[f] - \frac{1}{2^s} \sum_{y \in \{0,1\}^s} f(G(y)) \right| > \varepsilon \right] \leqslant \underbrace{2e^{-2\varepsilon^2 \cdot 2^s}}_{\text{tiny!}},$$

and by a union bound over all $f \in \mathcal{C}$, we have that

$$\Pr[G \text{ doesn't } \varepsilon\text{-fool } \mathcal{C}] \leqslant |\mathcal{C}| \cdot 2e^{-2\varepsilon^2 \cdot 2^s}.$$

Now if $s = \log\log|\mathcal{C}| + 2\log\frac{1}{\varepsilon} + O(1)$, then

$$2^s = \frac{|\mathcal{C}|}{\varepsilon^2} \cdot \underbrace{2^{-\log|\mathcal{C}|}}_{\text{tiny!}} \cdot \underbrace{2^{-2\log\frac{1}{\varepsilon}}}_{\text{tiny!}} \cdot \underbrace{2^{O(1)}}_{\text{tiny!}} \leqslant \frac{|\mathcal{C}|}{\varepsilon^2},$$

and therefore

$$\Pr[G \text{ doesn't } \varepsilon\text{-fool } \mathcal{C}] \leqslant \frac{|\mathcal{C}|}{\varepsilon^2} \cdot 2e^{-2\varepsilon^2 \cdot 2^s} \leqslant \frac{|\mathcal{C}|}{\varepsilon^2} \cdot 2e^{-2\varepsilon^2 \cdot \frac{|\mathcal{C}|}{\varepsilon^2}} = \frac{|\mathcal{C}|}{\varepsilon^2} \cdot 2e^{-2|\mathcal{C}|} \leqslant \frac{1}{2} < 1,$$

and so some outcome of G fools $\mathcal{C}$. □

In fact, there are natural constraints on the seed-length of a PRG: even the class $\mathcal{C}$ of all $\text{poly}(n)$-size circuits has $|\mathcal{C}| = 2^{\text{poly}(n)}$ and $S = O\left(\log\frac{n}{\varepsilon}\right)$. Lemma 1.15 shows that we can use randomness only once to get a PRG G, and that would indeed be enough. But still, we didn't eliminate randomness altogether; we didn't accomplish our goal of dining PRGs $G(n)$ that are computable in polynomial time via a deterministic algorithm with access to an s-bit string x. Fortunately this is not a problem, as, given a PRG, we can derandomise it by merely trying out all its seeds:

Lemma 1.16. *Suppose $G : \{0,1\}^s \to \{0,1\}^n$ is an explicit PRG that ε-fools $\mathcal{C}$. Then there exists a deterministic algorithm running in time $2^s \cdot \text{poly}(n)$ giving a value in $[\mathbb{E}[f] - \varepsilon, \mathbb{E}[f] + \varepsilon]$.*

The following corollary is not hard to show:

Corollary 1.17. *If we had an explicit PRG with seed-length s for the interesting class we have defined as $\mathcal{C} = \{$all $\text{poly}(n)$-size circuits$\}$, then we can get a $2^s \cdot \text{poly}(n)$-time deterministic algorithm for any language in* BPP.

Proof sketch. Given an input x to a randomised algorithm, view C_x as a circuit on random coins $r = (r_1, \ldots, r_n)$. Now fool C_x with a PRG G with seed-length s to get a circuit $C_{G(r)}$ that is close to C_x on all inputs x. We can then simply try all 2^s possible seeds r and check if $C_{G(r)}$ accepts x. □

From all our discussion, it follows that $O\left(\log\frac{n}{\varepsilon}\right)$ seed-length PRGs for polynomial-sized circuits would imply $\mathsf{BPP} = \mathsf{P}$. But we don't know how to construct such PRGs, and we don't know how to prove that they exist. This is our motivation for trying to find small seed-length PRGs for restricted classes of Boolean functions.

There is a natural connection between hardness and PRGs:

Lemma 1.18 (PRGs imply worst-case lower bounds)**.** *Let $\mathcal{C}$ be the class of n-bit functions such that $\mathcal{C}$ is closed under restrictions (i.e. if $f \in \mathcal{C}$ then the function evaluated on the input vector that has one of its components fixed to 0 or 1 is also in $\mathcal{C}$; note that all our classes $\mathcal{C}$ are closed under restrictions). Let $G : \{0,1\}^s \to \{0,1\}^n$ be an explicit ε-PRG for $\mathcal{C}$, where $s < n$ and $\varepsilon < 1/2$. Define the function $h : \{0,1\}^{s+1} \to \{0,1\}$ as that which outputs 1 on x if some string in the range of the PRG starts with x:*

$$h(x) = 1 \quad \textit{if and only if} \quad \exists\, y \in \{0,1\}^s, z \in \{0,1\}^{n-s-1} \quad \textit{such that} \quad G(y) = (x, z).$$

Let $\mathcal{C}'$ be all restrictions of functions in $\mathcal{C}$ by fixing the last $n - s - 1$ bits and leaving the first $s + 1$ bits alive. Then $h \notin \mathcal{C}'$.

Proof. G fools $\mathcal{C}$, and $\mathcal{C}$ is closed under restrictions. Therefore $G(\mathcal{U}_s)_{1,\ldots,s+1}$ ε-fools $\mathcal{C}'$. But G doesn't fool h; by the definition of h,

$$\mathbb{E}\left[h\left(G\left(\mathcal{U}_s\right)_{1,\ldots,s+1}\right)\right] = 1, \quad \text{while} \quad \mathbb{E}\left[h\left(\mathcal{U}_{s+1}\right)\right] = \Pr_{\boldsymbol{x}\sim\mathcal{U}_{s+1}}\left[h(\boldsymbol{x}) = 1\right] \leqslant \frac{1}{2},$$

because G has image size $\leqslant 2^s$. Therefore $h \notin \mathcal{C}'$. □

The significance of this result is that PRGs imply worst-case lower bounds, and we will see more about why this matters in the next lecture.

§1.4 Addendum: Complexity-theoretic vs cryptographic PRGs

Here we discuss the difference between complexity theoretic and cryptographic PRGs. Briefly, the high-level idea is the same in both settings: a PRG is a deterministic algorithm that stretches a short truly random seed (i.e. a uniform random bitstring) into a longer bitstring that "looks random" to a suitable observer. The difference is in the details: (1) what is the "suitable observer"; (2) how long is the longer bitstring; and (3) what is the required efficiency of the deterministic algorithm.

In the cryptographic setting:

1. The "suitable observer" is allowed to be any polynomial-time algorithm—i.e. we're fooling the class of all polynomial-time computations/polynomial-size circuits. (Of course, it's only known how to give explicit PRGs that do this using computational hardness assumptions!)
2. A PRG generally has "polynomial stretch": its input is an s-bit string, and its output is a string of length $\mathrm{poly}(s)$; and
3. The requirement on the deterministic algorithm is that it be polynomial-time computable.

For complexity-theoretic PRGs things are somewhat different:

1. In the context of proving unconditional results (where we don't make any computational hardness assumptions), the current state of the art is that we can only fool restricted classes that are weaker than all polynomial-size circuits. (For example, we'll see a couple of different unconditional PRGs against the class of all polynomial-size constant-depth circuits in this course.)
2. We think of a complexity-theoretic PRG as stretching an s-bit truly random seed to an n-bit output string, and we want the seed length to be as short as possible (because the point of having such a PRG is to enumerate over all 2^s seeds—that's how we approximate the expectation, under the uniform distribution, of the function that the PRG is fooling). So the idea here is to have n be a superpolynomially large function of s; ideally n would be about 2^s (equivalently, s would be about $\log n$). In the unconditional setting we'll sometimes be able to achieve this or something close to this; for example, for the class of polynomial-size constant-depth circuits, we'll get PRGs that have a seed length s which is $\mathrm{polylog}(n)$.
3. Finally, since as mentioned above the idea of having a PRG stretching s bits to n bits is that we'll cycle through all 2^s input strings to the PRG, it's fine for a complexity-theoretic PRG with seed length s to have running time $\mathrm{poly}(2^s, n)$ on its length-s input string.

§2 Lecture 02—23rd January, 2024

Last time.

- Overview of some restricted computational models we'll consider; worst-case, average-case lower bounds, and pseudorandom generators.
- PRGs imply worst-case lower bounds.

Today. Finish overview:

- Deterministic approximate counting.
- Boolean formulas: various worst-case lower bounds.

We start where we left off last time: PRGs imply worst-case lower bounds. We saw that if we had an explicit PRG with seed-length s for the class $\mathcal{C} = \{\text{all poly}(n)\text{-size circuits}\}$, then we can get a $2^s \cdot \text{poly}(n)$-time deterministic algorithm for any language in BPP. But we don't know how to construct such PRGs, and we don't know how to prove that they exist. This is our motivation for trying to find small seed-length PRGs for restricted classes of Boolean functions.

We will now show that PRGs imply average-case lower bounds. We will do this by showing that if we have a PRG G that fools a class $\mathcal{C}$ of functions, the likelihood that a random function $f \in \mathcal{C}$ is close to G is likely not more than half.

Lemma 2.1 (PRGs imply average-case lower bounds)**.** *Let $\mathcal{C}$ be a class of n-bit functions such that $\mathcal{C}$ is closed under restrictions. Let $G : \{0,1\}^s \to \{0,1\}^n$ be an ε-PRG for $\mathcal{C}$, where $s < n$ and $\varepsilon < 1/2$. Let $r = s + \log 1/\varepsilon \leqslant n$. Define the function $h : \{0,1\}^r \to \{0,1\}$ as that which outputs 1 on x if some string in the range of the PRG starts with x:*

$$h(x) = 1 \quad \textit{if and only if} \quad \exists\, y \in \{0,1\}^s, z \in \{0,1\}^{n-r} \quad \textit{such that} \quad G(y) = (x, z),$$

where x is an r-bit string and z is an $(n-r)$-bit string. Let $\mathcal{C}'$ be all restrictions of functions in $\mathcal{C}$ by fixing the last $n-r$ bits and leaving the first r bits alive. Let $\mathcal{D} = \frac{1}{2} \cdot \mathcal{U}_r + \frac{1}{2} G(\mathcal{U}_s)_{1:r}$. Then h is ε-hard for $\mathcal{C}'$ with respect to the distribution $\mathcal{D}$.

Proof. We have defined a suitable probability distribution $\mathcal{D}$ which is a uniform mixture of the uniform distribution over the truly random r-bit strings and the output of the pseudorandom generator on $G(\mathcal{U}_s)_{1:r}$. Let $\boldsymbol{u} \sim \mathcal{U}_r$ denote a sample from the truly random uniform distribution over s-bit strings, and let $\boldsymbol{u}' \sim \mathcal{U}_s$ denote a sample from the truly random uniform distribution over s-bit strings, where $r = s + \log 1/\varepsilon$.
Fix any $f \in \mathcal{C}'$, say $f(x) = f_0(x, a)$ for some fixed suffix $a \in \{0,1\}^{n-r}$ and some $f_0 \in \mathcal{C}$. Then

$$\begin{aligned}
\Pr_{\boldsymbol{x} \sim \mathcal{D}}[f(\boldsymbol{x}) = h(\boldsymbol{x})] &= \frac{1}{2} \Pr_{\boldsymbol{u} \sim \mathcal{U}_r}[f(\boldsymbol{u}) = h(\boldsymbol{u})] + \frac{1}{2} \Pr_{\boldsymbol{u}' \sim \mathcal{U}_s}\left[f\left(G(\boldsymbol{u}')_{1:r}\right) = h\left(G(\boldsymbol{u}')_{1:r}\right)\right] \\
&= \frac{1}{2} \Pr_{\boldsymbol{u} \sim \mathcal{U}_r}[f(\boldsymbol{u}) = h(\boldsymbol{u})] + \frac{1}{2} \Pr_{\boldsymbol{u}' \sim \mathcal{U}_s}\left[f\left(G(\boldsymbol{u}')_{1:r}\right) = 1\right],
\end{aligned}$$

from the way we defined h. Now it either holds that $f = h = 0$, or $f = h = 1$, and so we have that

$$\begin{aligned}
\Pr_{\boldsymbol{x} \sim \mathcal{D}}[f(\boldsymbol{x}) = h(\boldsymbol{x})] &= \frac{1}{2} \Pr_{\boldsymbol{u} \sim \mathcal{U}_r}[f(\boldsymbol{u}) = h(\boldsymbol{u})] + \frac{1}{2} \Pr_{\boldsymbol{u}' \sim \mathcal{U}_s}\left[f\left(G(\boldsymbol{u}')_{1:r}\right) = 1\right] \\
&\leqslant \frac{1}{2} \Pr_{\boldsymbol{u} \sim \mathcal{U}_r}[f(\boldsymbol{u}) = 0] + \frac{1}{2} \Pr_{\boldsymbol{u} \sim \mathcal{U}_r}[f(\boldsymbol{u}) = 1] + \frac{1}{2}\left(\Pr_{\boldsymbol{u} \sim \mathcal{U}_r}[f(\boldsymbol{u}) = 1] + \varepsilon\right),
\end{aligned}$$

since $f \in \mathcal{C}' \subseteq \mathcal{C}$ and G is an ε-PRG for $\mathcal{C}$. This then resolves to

$$\begin{aligned}\Pr_{\boldsymbol{x}\sim\mathcal{D}}[f(\boldsymbol{x}) = h(\boldsymbol{x})] &\leqslant \frac{1}{2} + \frac{1}{2}\left(\Pr_{\boldsymbol{u}\sim\mathcal{U}_r}[f(\boldsymbol{u}) = 1] + \varepsilon\right) \\ &= \frac{1}{2} + \frac{\varepsilon + \mathbb{E}[h]}{2} \leqslant \frac{1}{2} + \frac{2\varepsilon}{2} \\ &= \frac{1}{2} + \varepsilon,\end{aligned}$$

and similarly, we can show that $\Pr_{\boldsymbol{x}\sim\mathcal{D}}[f(\boldsymbol{x}) \neq h(\boldsymbol{x})] \geqslant \frac{1}{2} - \varepsilon$. Therefore $\mathrm{Cor}_{\mathcal{D}}[f, h] \geqslant 2\varepsilon$, and so h is ε-hard for $\mathcal{C}'$ under $\mathcal{D}$. □

§2.1 Deterministic approximate counting algorithms

PRGs are nice because they let us obliviously derandomise restricted classes of computation—the 2^s n-bit output strings of a PRG fool every function $f \in \mathcal{C}$ without looking at f itself, which is amazing! We are fighting an adversary (namely the class $\mathcal{C}$ of functions) without knowing anything in particular about the functions themselves. But what if we could do better by actually knowing f (for which we want to compute $\mathbb{E}[f] \pm \varepsilon$), we could use it by itself. This is the idea behind deterministic approximate counting algorithms.

Definition 2.2 (Deterministic approximate counting algorithm)**.** *Let $\mathcal{C}$ be a class of representations of functions $f : \{0,1\}^n \to \{0,1\}$. An algorithm is an ε-deterministic-approximate-counter (or ε-DAC) for $\mathcal{C}$ if it is deterministic, and on any input $F \in \mathcal{C}$, it outputs a value in the range $[\mathbb{E}[F(\mathcal{U})] - \varepsilon, \mathbb{E}[F(\mathcal{U})] + \varepsilon]$.*

Indeed it is not hard to show that given a poly-time ($\varepsilon = 0.1$)-DAC for the class $\mathcal{C}$ of all polynomial-sized Boolean circuits, then $\mathsf{P} = \mathsf{BPP}$.

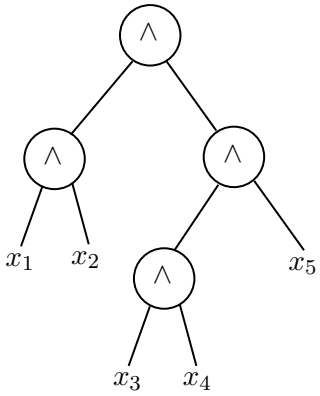
Fact 2.3. *For various classes $\mathcal{C}$, we can give ε-DACs which are faster than $2^s \cdot \mathrm{poly}(n)$, where s is the seed length of the best-known ε-PRG for $\mathcal{C}$.*

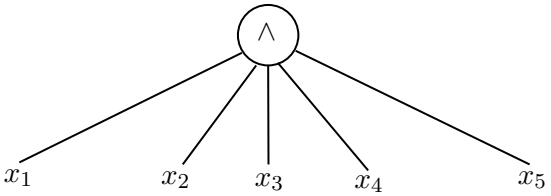
§2.2 Boolean formulas and their worst-case lower bounds

Definition 2.4 (Boolean formula)**.** *A* Boolean formula *(also called a U_2-basis or a $\{\wedge, \vee\}$-basis or a de Morgan formula) is a rooted binary tree with leaf nodes labelled with x_i or $\overline{x_i}$ for $i \in [n]$, and internal nodes labelled with $\wedge$ or $\vee$. The* size *of a Boolean formula is the number of leaves in the tree, and the* depth *of a Boolean formula is the length of the longest root-to-leaf path in the tree.*

Definition 2.5 (Formula size, depth)**.** *Given a function $f : \{0,1\}^n \to \{0,1\}$, the* formula size *of f is the minimum size of a Boolean representation F computing f. The* formula depth *of f is the minimum depth of a Boolean representation F computing f.*

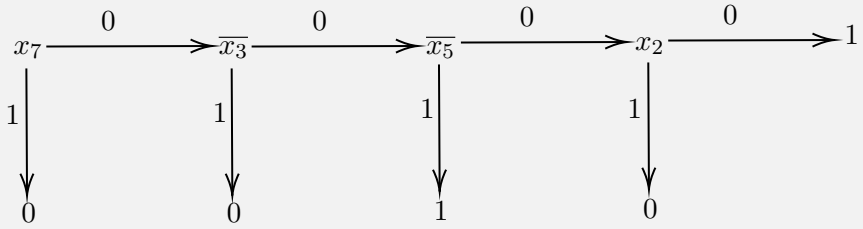
Sometimes we will consider Boolean formulas with unbounded fan-in; in these cases, the size of the Boolean formula unchanged but the depth changes; consider for example the two Boolean formulas





CNF and DNF formulas both have depth 2 and unbounded fan-in.

Example 2.6. The decision list function

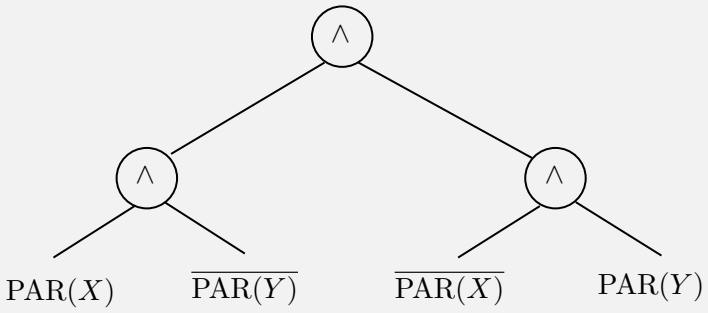


has the associated formula $\overline{x_7} \vee (x_3 \wedge (\overline{x_5} \vee \overline{x_2}))$.
Indeed, we can convert formulas with depth-2 unbounded fan-in to formulas with $2 \log n$ depth and $\Theta(n^2)$ size. We can do this by replacing each gate with a binary tree of depth $\log n$, and then replacing each leaf with a variable x_i or $\overline{x_i}$ with a tree of depth $\log n$.

Example 2.7. The parity function $\mathrm{PAR}_n : \{0,1\}^n \to \{0,1\}$ is

$$\mathrm{PAR}(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i \pmod 2.$$

The Boolean forula for this function is recursive: for $\mathrm{PAR}(x, y)$ where $x, y \in \{0,1\}^{n/2}$, we have that it is computed by the formula



Let $s(i)$ denote the size for i variables in this formula. Then we obtain the recurrence

$$\begin{cases} s(1) = 1 \\ s(2i) = 4s(i) \quad \text{for } i > 1. \end{cases}$$

Solving this recurrence, we get that $s(n) = \Theta(n^2)$.
This is exactly optimal when n is a power of 2! That said, we will obtain a lower bound of $\Omega(n^{1.5})$, using techniques relevant to PRGs.

Fact 2.8. *There are $O(n)$-size circuits for PAR_n.*

Why do we do better with Boolean circuits as opposed to formulas? The reason is that we can use the same gate multiple times in a circuit, but we cannot do so in a formula—circuits are more expressive than formulas and can reuse subcomputations.

§2.3 Shannon's (exponential) lower bound on formula size via counting arguments

We will now prove that there are functions which require exponential-size formulas. The idea here is very simple: every small formula has a concise description, but there are many many Boolean functions out there, so there aren't enough concise descriptions to go around—there's only a small number of small descriptions.

Fact 2.9. *There are 2^{2^n} many Boolean functions $f : \{0,1\}^n \to \{0,1\}$.*

Proof. There are 2^n many n-bit strings, and each of these can be mapped to either 0 or 1 via AND or OR, so there are 2^{2^n} many Boolean functions. □

Theorem 2.1 (Shannon's lower bound, [Sha49])**.** *A $1 - 2^{-n}$ fraction (actually much more) of all n-bit Boolean functions $f : \{0,1\}^n \to \{0,1\}$ have a formula size*

$$\mathrm{size}(f) \geqslant \frac{2^n}{2 \log n}.$$

Proof. We will show that for $s = 2^{n-1}(\log n)^{-1}$, the number of Boolean formulas of size s over Boolean variables $x_1, \ldots, x_n$ is $\leqslant 2^{-n} \cdot 2^{2^n}$. This will imply the theorem, since the number of Boolean functions is 2^{2^n}, and so the fraction of Boolean functions with formula size $< s$ is

$$\frac{2^{-n} \cdot 2^{2^n}}{2^{2^n}} = 2^{-n}.$$

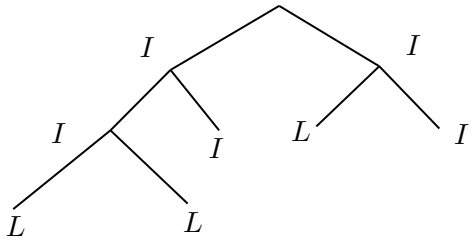
Here's why this is true: any formula f is specified by:

(I) the tree structure of the binary tree (essentially what it "looks like"), and

(II) the labelling of the leaves with x_i or $\overline{x_i}$.

We will show an upper bound on the number of both these specifications:

(I) An s-leaf formula has $s-1$ internal nodes. We can describe any such tree as an element of the alphabet $\{II, IL, LI, LL\}^{s-1}$, where II specifies that the first and second children are internal nodes, IL specifies that the first child is an internal node and the second child is a leaf, and so on. For example, the string $IIIILILL$ corresponds to the tree



Every tree has such a description, and so the number of trees with $s-1$ internal nodes is $\leqslant 4^{s-1}$.

(II) There are 2^{s-1} labellings for the $s-1$ internal nodes (either $\vee$ or $\wedge$), and there are $(2n)^s$ labellings for all the s leaves (either x_i or $\overline{x_i}$ for $i \in [n]$).

Therefore the total number of size-s formulas over $\{0,1\}$ is

$$\leqslant \underbrace{4^{s-1}}_{\text{trees}} \cdot \underbrace{2^{s-1}}_{\text{internal nodes}} \cdot \underbrace{(2n)^s}_{\text{leaves}} \leqslant (16n)^s.$$

For $s = 2^{n-1}(\log n)^{-1}$, we have that

$$(16n)^{2^n/(2\log n)} = 2^{2^n \cdot \frac{(\log n)+4}{2\log n}} \overset{\text{for sufficiently large } n}{\leqslant} 2^{0.51\cdot 2^n} \ll 2^{-n} \cdot 2^{2^n}. \qquad \square$$

Fact 2.10. *A DNF or CNF for f has $\leqslant 2^{n-1}$ terms and n literals per term, so we can easily get a worst-case lower bound of $n \cdot 2^{n-1}$ on the size of a DNF or CNF for f.*

§2.4 Subbotovskaya's lower bound on formula size via random restrictions

Subbotovskaya[1] was a Russian mathematician who analysed the effect of random restrictions on the size of Boolean formulas in her work [Sub61]. The idea is that given some Boolean formula F of size s computing a Boolean function $f(x_1, \ldots, x_n)$, we can fix some fraction of the variables to 0 or 1 and obtain a smaller formula F' computing f on the remaining variables. Essentially we will make the argument that setting some of the inputs to 0/1 values and leaving the rest free shrinks f "by a lot", and yet the size after the restriction is $\geqslant 1$, which would then mean that size(F) must have been large!

Definition 2.11 (Restriction)**.** *A restriction ρ is a mapping $[n] \mapsto \{0,1,*\}$ such that ρ is a partial assignment to variables $x_1, \ldots, x_n$, where $x_i \to *$ means that x_i is free/left unassigned.*
Equivalently, we can write $\rho \in \{0,1,*\}^n$. We write $f\restriction_{\rho}$ to denote the restriction of the function f to the variables specified by ρ.

[1] Unfortunately she lived a sad life and met a tragic end presumably at the hands of the KGB.

Example 2.12. Take $\rho = (1, 0, 1, *, *, 1)$ and a six-variable function f. Then $f\restriction_{\boldsymbol{\rho}}$ is the function $f(1, 0, 1, x_4, x_5, 1)$.

Effectively we can think of the restriction $f\restriction_{\boldsymbol{\rho}}$ as "zooming in" on a subcube of f.

Let $\mathcal{R}_k$ denote the following distribution over restrictions:

- choose k variables uniformly at random for $*$,
- let the other $n - k$ variables be fixed to 0 or 1 uniformly at random.

Theorem 2.2 (Subbotovskaya's lower bound, [Sub61])**.** *Let $f\colon \{0,1\}^n \to \{0,1\}$, and let $\boldsymbol{\rho}$ be drawn from $\mathcal{R}_k$. Then,*

$$\Pr_{\boldsymbol{\rho}\sim\mathcal{R}_k}\left[\operatorname{size}(f\restriction_{\boldsymbol{\rho}}) \leqslant 4 \cdot \left(\frac{k}{n}\right)^{3/2} \cdot \operatorname{size}(f)\right] \geqslant \frac{3}{4}.$$

Before we prove the theorem, let's see one of its applications. Let $k = 1$, and let $f = \mathrm{PAR}_n$. Then $f\restriction_{\boldsymbol{\rho}}$ is either x_i or $\overline{x_i}$, and so $\operatorname{size}(f\restriction_{\boldsymbol{\rho}}) = 1$. Therefore

$$1 \leqslant 4 \cdot \left(\frac{1}{n}\right)^{3/2} \cdot \operatorname{size}(f) \implies \operatorname{size}(f) \geqslant \frac{1}{4} \cdot n^{3/2}.$$

Proof of Theorem 2.2. Let F be an optimal (smallest) size Boolean formula for f, with $\operatorname{size}(F) = s$. View $\boldsymbol{\rho}$ as being constructed in $n - k$ stages, where at each stage we choose an unfixed variable uniformly at random and fix it to 0 or 1 uniformly at random. We'll analyse one of these stages. At the first stage, we pick x_i, and for some $b \in \{0, 1\}$, by fixing $x_i \to b$, all instances of x_i or $\overline{x_i}$ vanish. Then,

$$\mathbb{E}\left[\text{number of occupations of } x_i \text{ or } \overline{x_i}\right] = \frac{s}{n},$$

and consequently $\mathbb{E}[\text{size of } F]$ shrinks by s/n. The key to the proof is that F shrinks by more than s (in expectation). Consider any occurrence of x_i or $\overline{x_i}$ in F (perhaps in a $\wedge$ structure where there is an AND gate on top and branches out to x_i on the left and G on the right), where G is a subformula of size $\geqslant 1$. Note that G doesn't have x_i or $\overline{x_i}$: G only matters if $x_i = 1$, and then by replacing any occurrence of x_i or $\overline{x_i}$ in G with 1 or 0, we'd have a smaller formula, contradicting the optimality of F. Therefore with probability $1/2$, x_i gets fixed to 0, G then vanishes, and so there are $\geqslant 1$ literals vanishing. Therefore,

$$\mathbb{E}\left[\text{number of literals vanishing due to this secondary effect}\right] \geqslant \frac{1}{2} \cdot \frac{s}{n},$$

and consequently,

$$\begin{aligned}\mathbb{E}[\text{size of } F\restriction_{\boldsymbol{\rho}} \text{ after the first stage}] &\leqslant s - \frac{3}{2} \cdot \frac{s}{n} \\ &= \left(1 - \frac{3}{2n}\right) \cdot s \\ &\leqslant s \cdot \left(1 - \frac{1}{n}\right)^{3/2}.\end{aligned}$$

Now we repeat; on the second step with $n-1$ variables, so we have

$$\begin{aligned}\mathbb{E}[\text{size of } F\restriction_{\boldsymbol{\rho}} \text{ after the second stage}] &\leqslant s \cdot \left(1-\frac{1}{n}\right)^{3/2} \cdot \left(1-\frac{1}{n-1}\right)^{3/2} \\ &= s \cdot \left(\frac{n-1}{n}\right)^{3/2} \cdot \left(\frac{n-2}{n-1}\right)^{3/2},\end{aligned}$$

and therefore after $n-k$ stages,

$$\begin{aligned}\mathbb{E}[\text{size of } F\restriction_{\boldsymbol{\rho}} \text{ after } n-k \text{ stages}] &\leqslant s \cdot \left(\frac{n-1}{n}\right)^{3/2} \cdot \left(\frac{n-2}{n-1}\right)^{3/2} \cdots \left(\frac{k}{k+1}\right)^{3/2}, \\ &= s \cdot \left(\frac{k}{n}\right)^{3/2}.\end{aligned}$$

Then by Markov's inequality, with probability $\geqslant 3/4$,

$$\text{size}(F\restriction_{\boldsymbol{\rho}}) \leqslant 4 \cdot \left(\frac{k}{n}\right)^{3/2} \cdot \text{size}(F). \qquad \square$$

Call Γ the shrinkage exponent of f if for any restriction ρ, i.e. the Γ for which

$$\Pr_{\boldsymbol{\rho}\sim\mathcal{R}_k}\left[\text{size}(f\restriction_{\boldsymbol{\rho}}) \leqslant 4 \cdot \left(\frac{k}{n}\right)^{\Gamma} \cdot \text{size}(f)\right] \geqslant \frac{3}{4}.$$

It is known that $\Gamma \leqslant 2$ is an upper bound on the shrinkage exponent of any Boolean function f. There is a body of active research trying to establish a lower bound on the shrinkage exponent: Nisan and Impagliazzo [IN93] showed that $\Gamma \geqslant 1.55$, Paterson and Zwick [PZ93] showed that $\Gamma \geqslant 1.63$, Håstad [Has93] showed that $\Gamma \geqslant 2-o(1)$, and Avishay Tal [Tal14] gave a better $\Gamma \geqslant 2-o(1)$ bound via spectral techniques.

§2.5 Andre'ev's lower bound

This step due to Andre'ev is a bound that provides a clever way to get an $n^{\Gamma+1}$ formula size lower bound for an explicit function. Since we proved that $\Gamma = 3/2$ via Theorem 2.2, we get a lower bound of $n^{5/2}$ for an explicit function for free. If we take Avishay Tal's result as fact, then we get a lower bound of $n^{3-o(1)}$ for an explicit function.

Warmup. We saw that there is an n-variable function with formula size $\geqslant 2^n/(2\log n)$. Can we scale down? Yes! We can scale down to, say, $b = \lg n$ variables, and what this tells us is that there is some $(\lg n)$-variable function requiring formula size

$$\text{size}(F) \gtrapprox \frac{n}{\log\log n}.$$

Call this function $\psi(x_1,\ldots,x_n)$. Let $m = n/b = n/\log n$, and view $x_1,\ldots,x_n$ as b blocks of m variables per block. Define $f_\psi\colon \{0,1\}^m \to \{0,1\}$ as follows:

$$f_\psi(x_1,\ldots,x_n) = \psi\left(x_1 \oplus \cdots \oplus x_m, x_{m+1} \oplus \cdots \oplus x_{2m}, \ldots, x_{n-m+1} \oplus \cdots \oplus x_n\right).$$

Now apply Subbotovskaya's lower bound (Theorem 2.2) and the shrinkage exponent Γ to the function f_ψ with $k = b \cdot \ln(4b)$, so that $\boldsymbol{\rho} \sim \mathcal{R}_k$. Then we have that with probability $\geqslant 3/4$,

$$\operatorname{size}(f_\psi\restriction_{\boldsymbol{\rho}}) \leqslant 4 \cdot \left(\frac{k}{m}\right)^{\Gamma} \cdot \operatorname{size}(f_\psi).$$

The following claim is useful for our argument:

Claim 2.13. *With probability $\geqslant 3/4$ over $\boldsymbol{\rho} \sim \mathcal{R}_k$, we have that $\boldsymbol{\rho}$ fixes $\geqslant 1 *$ to each block.*

Proof. It's a straightforward calculation:

$$\begin{aligned}\Pr_{\boldsymbol{\rho}}[\text{block } i \text{ gets no } * \text{ under } \boldsymbol{\rho}] = \frac{\binom{n-m}{k}}{\binom{n}{k}} &\leqslant \left(\frac{n-m}{n}\right)^k \\ &= \left(1 - \frac{1}{b}\right)^{b \cdot 4 \ln b} \leqslant \frac{1}{4b}.\end{aligned}$$

Then by the union bound over all b blocks, the probability that any block gets no $*$ is $\leqslant 1/4$, and so the probability that any block gets $\geqslant 1 *$ is $\geqslant 3/4$. □

Therefore with probability $\geqslant \left(1 - \frac{3}{4}\right) + \left(1 - \frac{3}{4}\right)$, the events in both Claim 2.13 and Theorem 2.2 overlap, and so with probability $\geqslant 1/2$ the restriction $\boldsymbol{\rho} \sim \mathcal{R}_k$ satisfies both Claim 2.13 and Theorem 2.2. Fix any such $\boldsymbol{\rho}$. For such a $\boldsymbol{\rho}$, we have that $f_\psi\restriction_{\boldsymbol{\rho}}$ has ψ as a subfunction, and therefore,

$$\operatorname{size}(f_\psi\restriction_{\boldsymbol{\rho}}) \geqslant \operatorname{size}(\psi) \geqslant \frac{n}{\log\log n}.$$

Putting the pieces together,

$$\begin{aligned}\frac{n}{\log\log n} \leqslant \operatorname{size}(\psi) &\leqslant \operatorname{size}(f_\psi\restriction_{\boldsymbol{\rho}}) \\ &\leqslant 4 \cdot \left(\frac{k}{m}\right)^{\Gamma} \cdot \operatorname{size}(f_\psi) \\ &= 4 \cdot \left(\frac{(\log n) \cdot \ln(4 \ln n)}{n}\right)^{\Gamma} \cdot \operatorname{size}(f_\psi),\end{aligned}$$

which means that

$$4 \cdot \left(\frac{(\log n) \cdot \ln(4 \ln n)}{n}\right)^{\Gamma} \cdot \operatorname{size}(f_\psi) \geqslant \frac{n}{\log\log n}.$$

So $\operatorname{size}(f_\psi) = \widetilde{\Omega}\left(n^{\Gamma+1}\right)$.

However, this f_ψ is not explicit! Here's a trick we can use to make it explicit. Give f_ψ the ψ as input, and view $\psi(x_1, \ldots, x_b)$ as a $(2^b = n)$-bit string. Then for any $y \in \{0,1\}^n$, view f_y as a b-bit function:

$$f_y(x_1, \ldots, x_m) = y\left(x_1 \oplus \cdots \oplus x_m, x_{m+1} \oplus \cdots \oplus x_{2m}, \ldots, x_{n-m+1} \oplus \cdots \oplus x_n\right).$$

Then f_y is a $2n$ variable function and is explicit; it is in fact Andre'ev's function on n-bit inputs x and y. Some $y \in \{0,1\}^n$ gives ψ, and so

$$\operatorname{size}\left(A(x,y)\right) \geqslant \operatorname{size}(f_\psi) \geqslant \widetilde{\Omega}\left(n^{\Gamma+1}\right).$$

The Andre'ev's function A is computable in polynomial time by a linear-sized circuit; it is in P. Indeed A has circuit size $O(n)$ and formula size $\widetilde{\Omega}(n^3)$.

§3 Lecture 03—30th January, 2024

Last time. In the previous class, we

- Finished the last bit of overview; showed that if we have a PRG for a class $\mathcal{C}$, we not only get worst-case lower bounds, but also average-case lower bounds. (And so if we have a class of functions in a restricted computational model, we should try to first prove worst-case lower bounds; if we succeed, we should then try to prove average-case lower bounds, and if we succeed, we can try to give PRGs for that class.)
- Mentioned deterministic approximate counting—which gets us to the same endpoint of figuring out how many satisfying assignments there are for these different types of functions, but via an algorithmic route.
- Gave various worst-case lower bounds for Boolean formulas:
  - Shannon: non-explicit $\Omega(2^n/\log n)$ lower bound;
  - Subbotovskaya: lower bound of $\Omega(n^{1.5})$ for the parity function PARITY;
  - Andre'ev: lower bound of $\Omega(n^{2.5})$ for $A(x,y) = f_y(x)$, where f_y is the function

$$f_y(x_1,\ldots,x_m) = y\left(\bigoplus_{i=1}^{m} x_i, \bigoplus_{i=m+1}^{2m} x_i, \ldots, \bigoplus_{i=n-m+1}^{n} x_i\right).$$

Today. Finish overview:

- Deterministic approximate counting.
- Boolean formulas: various worst-case lower bounds.

The readings for today are Sections 6.1, 12.1, and 12.2 of [J+12] and Sections 3.3 and 3.4 of [BS90].

§3.1 The Karchmer-Raz-Widgerson conjecture

Last time we proved a lower bound of $\Omega(n^{2.5})$ for the Andre'ev function $A(x,y)$ formula size, if we believe that the shrinkage exponent $\Gamma = 2$. The following question is natural:

Question: Can we do better than this, perhaps by recursively applying the reasoning we used in the construction of a hard function for an augumented version of the Andre'ev function?
▷ Suppose we take an input of size $2n\log n + n$ which we view as $f(x_1,\ldots,x_{\log n},\psi)$, where all the x_i have length $2n$ and we interpret the last n bits as encoding a function on $\log n$ variables. However, the proof doesn't work via Andre'ev's methods.

The KRW conjecture is a famous conjecture related to this question of whether the formula size behaves as we expect it to for these simple combinations.

Conjecture 3.1 (Karchmer-Raz-Wigderson (KRW), [KRW95])**.** *Given two Boolean functions* $f:\{0,1\}^n \to \{0,1\}$ *and* $g:\{0,1\}^m \to \{0,1\}$*, write* $g \circ f:\{0,1\}^{nm} \to \{0,1\}$ *to denote the* disjoint

composition

$$(g \circ f)(x_1, \ldots, x_{nm}) = g\left(f(x_1, \ldots, x_n), f(x_{n+1}, \ldots, x_{2n}) \ldots, f(x_{nm-n+1}, \ldots, x_{nm})\right).$$

It is clear that we can get a formula upper bound for the depth of this combined function given formula upper bounds for the depth of f and g:

$$\text{depth}(g \circ f) \leqslant \text{depth}(f) + \text{depth}(g).$$

The KRW conjecture *claims that this is the best we can do; that is, for any two nontrivial functions f and g, the formula depth of $g \circ f$ is more or less the same as the sum of the formula depths of f and g:*

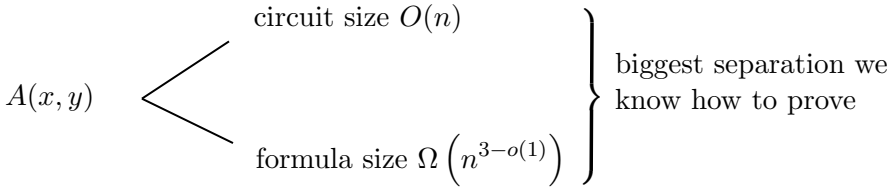
$$\text{depth}(g \circ f) \cong \text{depth}(f) + \text{depth}(g).$$

There's been a lot of progress on proving certain special cases of this conjecture, but it remains open in general. If it were true, we would get very strong formula size lower bounds (and super-polynomial circuit lower bounds), because there is a close connection between formula depth and size in general as we will see next.

Question: What does $\cong$ mean in the statement of the conjecture?
$\triangleright$ The statement of the KRW conjecture is a bit vague and can be formulated in different ways depending on how much precision is desired. For instance, we could say that "$\text{depth}(g \circ f) \cong \text{depth}(f) + \text{depth}(g)$" means $\text{depth}(g \circ f) = \Theta(\text{depth}(f) + \text{depth}(g))$. The point is that the formula depth of $g \circ f$ is (claimed to be) not much smaller or much larger than the sum of the formula depths of f and g.

§3.2 Depth versus size for Boolean formulas

We already saw that Andre'ev's function A has

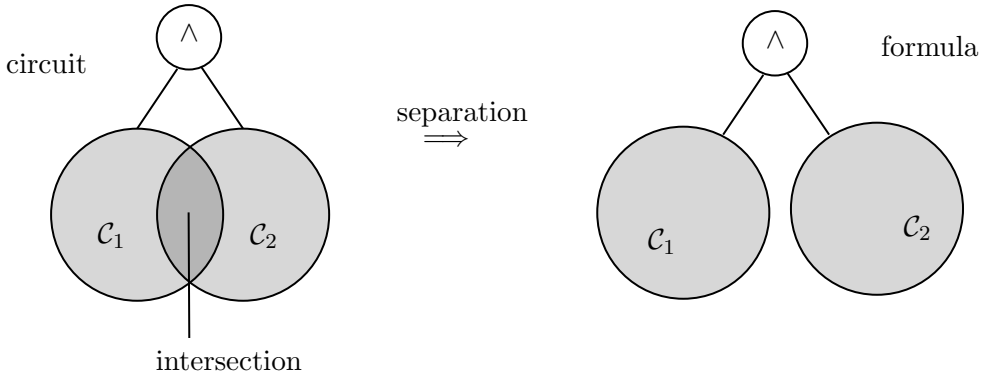


if we believe that the shrinkage exponent is $2 - o(1)$[2]. But if we're interested in depth, there is no gap for any function!

Claim 3.2. *If a Boolean function has a fanin-2, depth-d $\{\wedge, \vee, \neg\}$ circuit, then f has a fanin-2, depth-d $\{\wedge, \vee, \neg\}$ formula of size 2^d (and the converse also holds for the depth).*

Proof. Trivially, if a function f has a fanin-2 formula of depth d, then f has a fanin-2 circuit of depth d, since every formula is a circuit. For the forward direction, we can prove the claim by induction on the depth of the circuit.

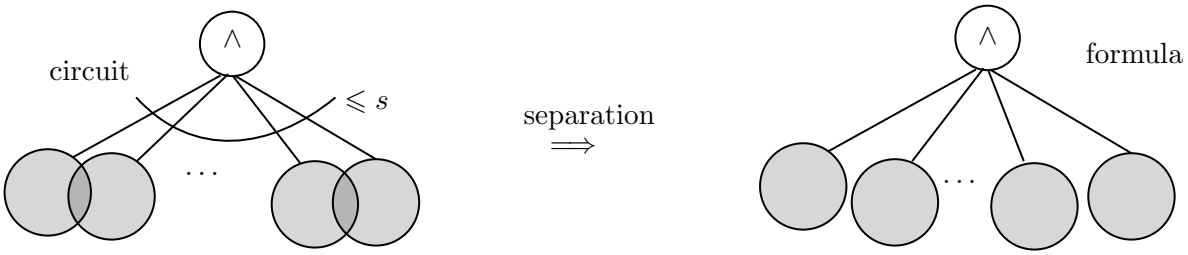
[2]We believe there are bigger separations; we believe that $\mathsf{P} \neq \mathsf{NC}^1$, and so we believe that there are functions which have polynomial-sized circuits but do not have superpolynomial-sized formulas, but we don't know how to prove this.



If f is a single variable, then f has a fanin-2 formula of depth 1 and size 1. Suppose that f has a fanin-2, depth-$(d-1)$ circuit. Then we can write f as $f = \mathcal{C}_1 \wedge \mathcal{C}_2$, where $\mathcal{C}_1$ and $\mathcal{C}_2$ are fanin-2, depth-$(d-1)$ circuits. By the inductive hypothesis, $\mathcal{C}_1$ and $\mathcal{C}_2$ have fanin-2, depth-$(d-1)$ formulas of size 2^{d-1}. Then following the picture, f has a fanin-2, formula of depth $d-1+1 = d$ and size $2^{d-1} + 2^{d-1} = 2^d$. □

Related claim. If f has an unbounded fan-in circuit of depth d and size s, then f has an unbounded formula of depth d and size s^d, and the converse also holds for the depth.

Proof. The proof is similar to that of the previous claim:



Just apply the same inductive argument as before. □

Here's a natural question:

Question: Do we really need this exponential blowup in size, or is there a smarter way to avoid it by any chance?
▷ In some settings, yes. Rossman [Ros14] showed that for some functions and for some sufficiently small d, there exist size-s depth-d circuits with no size $s^{o(d)}$ depth-d formula.

So what *do* we know about the relationship between formula depth and formula size?

Theorem 3.1 ([Spi71])**.** *For any Boolean function f,* $\text{depth}(f) = \Theta(\log \text{size}(f))$.

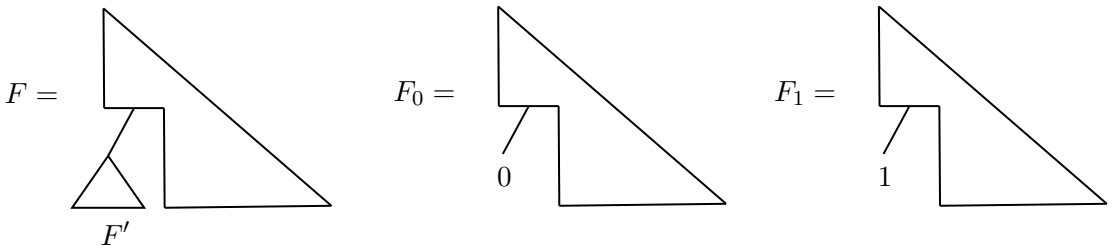
Proof. One direction is easy: given F a Boolean formula, we know that it is a binary tree by definition, and for every binary tree, the number of leaves is at least $2^{\text{depth}(F)}$. So $\text{size}(F) \leqslant 2^{\text{depth}(F)}$, and so $\log \text{size}(F) \leqslant \text{depth}(F)$. This holds for any formula, so it must hold for the optimal formula, and therefore $\log \text{size}(F) \leqslant \text{depth}(F)$ for any function F.

For the other direction, we want to show that $\text{depth}(F) \leqslant O(\log \text{size}(F))$. The following lemma will be key for us:

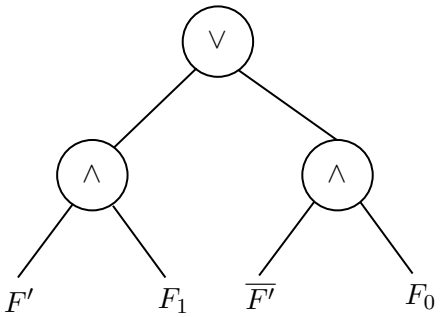
Lemma 3.3. *Let T be a rooted binary tree with s leaves. Then T has some subtree with s' leaves for some $s' \in [s/3, 2s/3]$.*

Proof. From a tree of size s, pick a particular node; this splits the tree into subtrees of size r and $s-r$. One of these is at least $s/2$. Pick that one and repeat this strategy of splitting it until we get a subtree in the range $[s/3, 2s/3]$. This has to work, because a tree of size at least $2s/3$ cannot be split into two subtrees both less than $s/3$. So we have a subtree of size s' in the range $[s/3, 2s/3]$. □

Continuation of the proof of Theorem 3.1. Now fix an optimal formula F of size s for F. Let F' be its subformula of size $s' \in [s/3, 2s/3]$ (which exists via the lemma above). Let the subformula F_0 be F but with F' replaced by 0, and let the subformula F_1 be F but with F' replaced by 1.



Note that $\text{size}(F'), \text{size}(F_0), \text{size}(F_1) \in [s/3, 2s/3]$, because F' is a subformula of F, and F_0 and F_1 are merely modifications of F as it relates to F'. Now by Boolean logic, we have that $F = (F' \wedge F_1) \vee (\overline{F'} \wedge F_0)$, and the following is a formula for F:



In essence, we have "balanced" the formula F by replacing F' with 0 in one part and 1 in the other, and gained 2 more in depth by this process, since we can readily check that

$$\text{depth}(F) \leqslant 2 + \max\{\text{depth}(F'), \text{depth}(F_0), \text{depth}(F_1)\}.$$

Recursively apply the same balancing procedure to each of F', F_0, and F_1, and define $D(s)$ to be the maximum possible depth obtainable from this procedure over all size $\leqslant s$ formulas. Then we have that $D(s) \leqslant 2 + D(2s/3)$, and so

$$D(s) \leqslant O(\log s) = O(\log \text{size}(F)),$$

which is what we set out to prove. □

§3.3 Full-basis formulas and circuits

Until now, our discussion has been focused on Boolean formulas in the de Morgan basis with $\wedge, \vee$, and $\neg$ operations. However, *full-basis* formulas allow any two-variable binary gate $g : \{0,1\}^2 \to \{0,1\}$. It is easy to show that full-basis formulas are the same in power as gates that use only binary XOR ($\oplus$) and AND ($\wedge$) gates:

Claim 3.4. *Given a full-basis formula with ℓ variable leaves, there always exists an equivalent formula with ℓ variable leaves using only binary XOR ($\oplus$) and AND ($\wedge$) gates.*

Proof. Note that we can obtain all the binary functions with two variable leaves:

1. Negating variables, i.e. $\neg x_i \equiv x_i \oplus 1$.
2. Gates where two (out of four) of the input combinations map to 1's are either a single variable or the binary XOR function (or their negations).
3. Gates where (one or three, by way of negation) of the input combinations map to 1 can be constructed in the following way:

x	y	$\overline{x} \wedge \overline{y}$	$\overline{x} \wedge y$	$x \wedge \overline{y}$	$x \wedge y$
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

So we can replace any full-basis formula with an equivalent formula using only binary XOR ($\oplus$) and AND ($\wedge$) gates. □

What else happens in this model? Do we need to reprove everything? Fortunately, no:

- Shannon's $\Omega(2^n/\log n)$ bound still holds, as it was obtained via a counting argument on Boolean formulas;
- The $\Omega(n^{1.5})$ lower bound for PARITY_n breaks down—we have full-basis formula size n for PARITY_n since we can construct the tree of parities we saw in a previous lecture and compute PARITY_i on all the i variables at each level.

In fact, Nečiporuk [Nec66] showed that a full basis formula for an explicit function over n variables has size $\Omega(n^2/\log n)$. We will prove a looser version of this result via Andre'ev-like method (but without using random restrictions).

Let $b = \log n$ and let $m = n/b = n/\log n$. Define a $2n$-variable function

$$\begin{aligned} A(x,y) &= A\left(x_1, x_2, \ldots, x_m, y_1, y_2, \ldots, y_n\right) \\ &= \underbrace{f_y(x_1 \wedge \ldots \wedge x_m, x_{m+1} \wedge \ldots \wedge x_{2m}, \ldots)}_{b \text{ blocks, } m \text{ variables per block}}. \end{aligned}$$

Claim 3.5. *Any full-basis formula for $A(x, y)$ must have size*

$$\geqslant \Omega\left(\frac{n^2}{\log n \log\log n}\right).$$

Proof. Let F be a minimum size formula for A with $\text{size}(F) = s$. Pick $\log n$ blocks of $n/\log n$ variables each thus: for $m = n/\log n$, define

$$\begin{aligned} B_1 &= x_1 x_2 \cdots x_m, \\ B_2 &= x_{m+1} x_{m+2} \cdots x_{2m}, \\ &\vdots \\ B_{\log n} &= x_{(n-1)m+1} x_{(n-1)m+2} \cdots x_{m \log n}. \end{aligned}$$

Now select the least frequently occurring variable in each block (without loss of generality, say x_m in B_1, x_{2m} in B_2, and so on), and let restriction $\boldsymbol{\rho}$ be that which fixes all other variables equal to 1. This causes each block B_i to collapse to x_{im}, and so $A\restriction_{\boldsymbol{\rho}}$ is equivalent to $f_y(x_m, x_{2m}, \ldots, x_{bm})$. Nowe we can pick $y \in \{0,1\}^n$ so that f_y is any $(\log n)$-variable function. By Shannon's counting argument, we have that most f_y will require size $\Omega(2^{\log n}/\log\log n) = \Omega(n/\log\log n)$. Then F will have $\geqslant \Omega(n/\log\log n)$ occurrences of the variables $x_m, x_{2m}, \ldots, x_{bm}$, and so the total number of occurrences of $x_1, \ldots, x_n$ in F must be

$$\geqslant m \cdot \Omega(n/\log\log n) = \Omega\left(\frac{n^2}{\log n \log\log n}\right),$$

as desired. □

Question: Given an arbitrary (potentially non-binary) complete basis, does the same thing hold?
▷ No; there indeed exist complete bases that do not trivially convert to something like what we have seen, or are not equivalent to a de Morgan-type basis.

Some other known lower bounds for full-basis formulas include the following:

- The best lower bound for MAJ_n is $\Omega(n \log n)$ full-basis formula size (proved via an argument based on Ramsey theory);
- An argument via multi-party communication complexity gives $\Omega(n \log^2 n)$ for any explicit function.

These are both non-Andre'ev/non-Nečiporuk methods, and a bit involved.

§3.4 Constant-depth circuits

Constant-depth circuits (which we will call AC^0 or $\mathsf{AC}^0_{d,s}$ for a (depth, size) pair (d, s)) are a natural computation model in which the standard gates have unbounded fanin (i.e. can take any number of inputs). A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is in AC^0 if there exists a polynomial $p(n)$, a constant d, and a circuit family $\{\mathcal{C}_n\}_{n=1}^{\infty}$ for the function f over $\wedge, \vee, \neg$ gates of unbounded fanin such that $\mathcal{C}_n$ is of size at most $p(n)$ and depth at most d (considered to be small in some sense; either constant or a slow-growing function of n).[3]

[3] Note that DNFs and CNFs are the $d = 2$ cases of $\mathbf{AC}^0$.

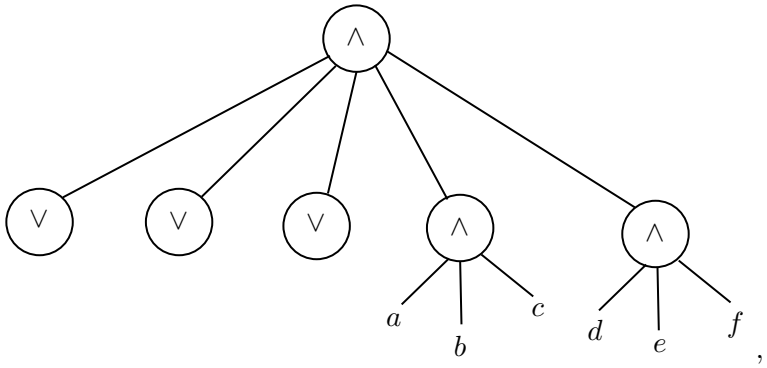
Motivation. We can think of $\mathsf{AC}^0_{d,s}$ as a very general parallel computation model. Each level of the circuit can be thought of as a timestep, and thus a depth-d constant-depth circuit is like d-timesteps on processors that can only perform $\wedge, \vee, \neg$ operations. Sufficiently strong lower bounds for constant-depth circuits would also imply nontrivial lower bounds for more general circuit models—a classic example is due to Valiant:

Theorem 3.2 (Valiant, [Val77]). *If a Boolean function $f : \{0,1\}^n \to \{0,1\}$ requires any depth-3 circuit to have size $2^{\omega\left(\frac{n}{\log\log n}\right)}$, then f cannot be computed by a circuit that has both $O(n)$ size and $O(\log n)$ depth.*

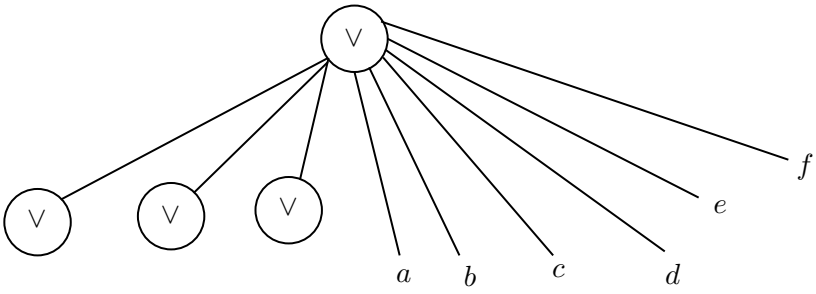
Strong enough lower bounds for depth-3 will hopefully tell us something nontrivial about richer models with logarithmic depth and linear size—although we don't have any of those results yet.

Let's now make two simplifying observations that help us clarify our thinking about AC^0:

- Since d is small in some sense, we can view our size-s depth-d circuit as a size-s^d depth-d formula (via the claim we already saw) without suffering too much blowup. So we can consider only formulas instead, which are somewhat simpler because they don't reuse subcomputations.
- Given a formula, we can assume all negations are at the leaves, and that gates alternate between $\wedge$ and $\vee$. This is because it never makes sense to have a formula like



since it is equivalent to



no matter what the leaf variables are.

§3.4.1 Intuition for the lower bound of constant-depth circuits

We will now give a high-level intuition for the lower bound of constant-depth circuits. We start by presenting function $f : \{0,1\}^n \to \{0,1\}$ that is plausibly "hard" for constant-depth circuits of

depth $d = 2$. With some thought, we find that the parity function PARITY_n is the ultimate hard function for DNFs.

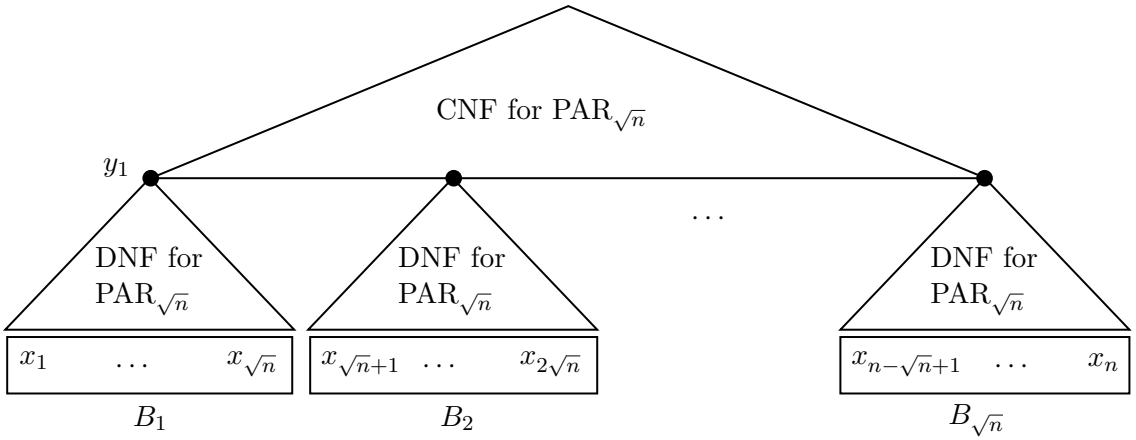
Claim 3.6. *Any (optimal) DNF $\mathcal{D}$ for PARITY_n must have 2^{n-1} AND gates, and in fact, we must have one AND gate for each of the 2^{n-1} satisfying assignments of the input variables.*

Proof. Suppose the existence of some AND gate $\mathcal{A}$ in the DNF $\mathcal{D}$ that is unaffected by input i. As $\mathcal{D}$ is optimal, $\mathcal{A}$ is not identically 0, and thus there must exist some $x \in \{0,1\}^n$ such that $\mathcal{A}(x) = 1$, and $\mathcal{A}(y) = 1$ for all y that differ from x only in the i-th bit. But then $\mathcal{D}(x) = \mathcal{D}(y) = 1$ for all $y \in \{0,1\}^n$, and so $\mathcal{A}$ doesn't actually compute the parity function.

Now we show that $\mathcal{D}$ has at least 2^{n-1} AND gates. As every AND gate contains all the n variables, each gate ust be satisfied by no more than 1 input string. To compute the parity function, we need at least one AND gate to be satisfied for each of the 2^{n-1} strings in $\text{PARITY}_n^{-1}(1)$. Thus, $\mathcal{D}$ must have at least 2^{n-1} AND gates. □

For example, if $n = 5$ and the DNF $\mathcal{D}$ has $x_1\overline{x_2}x_3\overline{x_4}$ as a term; it accepts 10100 and 10101, but PARITY_n is 1 on the former and 0 on the latter. The idea is then that each term in any DNF (especially an optimal one) must have length at most n and will accept exactly one satisfying assignment, and so we need 2^{n-1} terms. By the same argument, we can show that any (optimal) CNF for PARITY_n must have 2^{n-1} OR clauses, and more generally, we have DNFs and CNFs for PARITY_n of size 2^{n-1} and depth 2.

What about depth-3 circuits? As it turns out, we can compute PARITY_n with a depth-3 circuit of size $\approx 2^{\sqrt{n}}$, and we will show that this is true via a divide-and-conquer type approach.



Divide n into $\sqrt{n}$-many $\sqrt{n}$-sized blocks, and:

- Use a DNF of size $\approx 2^{\sqrt{n}}$ to compute the parity $\text{PARITY}_{\sqrt{n}}$ on each block to obtain outputs $y_1, \ldots, y_{\sqrt{n}}$;
- Use a CNF[4] to compute $\text{PARITY}(y_1, \ldots, y_{\sqrt{n}})$, and by so doing, effectively collapse the two layers of adjacent OR gates into a single layer, and so we have a circuit of size $(\sqrt{n}+1) \cdot 2^{\sqrt{n}}$ and depth 3.

[4] If we use a DNF for $\text{PARITY}(y_1, \ldots, y_{\sqrt{n}})$, we would get the total circuit size as about $(\sqrt{n}+1) \cdot 2^{\sqrt{n}}$ and circuit depth 4. This has an extra level of depth, so it's not the smartest choice for us.

Indeed it is not hard to generalise this idea:

Claim 3.7. *There exists a circuit of size* $\approx 2^{O\left(n^{1/(d-1)}\right)}$ *and constant-depth* d *for* PARITY_n*, and in fact, this is the best possible upper bound obtainable for* PARITY_n*.*

Proof. Left as a homework exercise. □

§3.4.2 Key to Håstad's lower bound: Håstad's switching lemma

Now the proof of the upper bound is not hard; that of the lower bound proved in the 1980s, however, is one of the crowning jewels of complexity theory and is much less obvious. The main theorem due to Håstad is the following:

Theorem 3.3 (Håstad, [Has86])**.** *For* $d \lessapprox \frac{\log n}{\log \log n}$*, any* $\mathsf{AC}^0_{d,s}$ *circuit for* PARITY_n *must have size* $2^{\Omega\left(1/\left(n^{d-1}\right)\right)}$.

From the 1980s, there was a lot of work focused on finding bounds on small depth circuits for functions such as PARITY. Some of the major results progressed as follows:

- Ajtai [Ajt83], Furst, Saxe, and Sipser [FSS84] proved that $\text{PARITY} \notin \mathsf{AC}^0$.
- Yao [Yao85] proved the first exponential lower bounds on parity via combinatorial methods, but Håstad introduced an entirely new technique, his *switching lemma*, to prove even stronger lower bounds. Håstad's method relies on purely combinatorial and probabilistic techniques.
- In the same decade, Razborov and Smolensky were introducing algebraic techniques to prove new results for polynomials over finite fields. Smolensky in particular showed [Smo87] that depth-d circuits with gates AND, OR, and MOD_p, where p is a prime, require at least $2^{n^{\Omega(1/(2d))}}$ gates to calculate MOD_r for any $r \neq p^m$. Yao's parity result is essentially a special case of this result.

Before we present the switching lemma, why *could* Theorem 3.3 be true? We understand the situation clearly for $d = 2$: CNFs and DNFs need large circuits to compute parity. If we could make a depth d circuit into a CNF/DNF without too much size blowup, then we'd 'beat' Theorem 3.3. If the bottom two layers of the circuit look like a DNF (i.e. we have $\vee$ then $\wedge$ gates), and we could efficiently switch it to a CNF (i.e. $\wedge$ then $\vee$ gates), then we'd be in business, because the circuit would then collapse one level (via our assumption that the levels alternate between $\wedge$ and $\vee$). If we can do this without paying too much in size, we can essentially repeat the process $\approx d$ times and get a small DNF/CNF for parity (and thus beat Theorem 3.3).

So in some sense our question is the following:

> Can we convert the constant-depth circuit $\mathcal{C}$ for PARITY into a DNF/CNF for PARITY, without increasing its size too much?

If so, then we are done, by our very strong DNF/CNF lower bounds. The problem though is that a CNF with n terms can blow up to a DNF with $2^{n/2}$ terms. For example, consider the CNF

$$(x_1 \vee x_2) \wedge (x_3 \vee x_4) \wedge \cdots \wedge (x_{2k-1} \vee x_{2k}).$$

By de Morgan's laws, the DNF form of this CNF would take at least 2^k terms, and hence there is exponential blowup. Our goal is then to retain the same general ideas of the above approach, but to do so in a way that avoids this blowup. For this, we go back to random restrictions. If we randomly restrict some of the variables to take on fixed values, say 0 or 1 only, then we might imagine that a lot of the terms will become much simpler and smaller in size, and therefore might possibly lead to a DNF conversion that doesn't incur an exponential blowup. So we have hope: a random restriction "kills off" longer terms, so using random restrictions can help us convert small CNFs to small DNFs.

For $0 < p < 1$ the survival probability of a Boolean variable, let $\mathcal{R}_p$ be a distribution over restrictions $\boldsymbol{\rho} : [n] \to \{0, 1, *\}$ such that

$$\boldsymbol{\rho}(x_i) = \begin{cases} * & \text{with probability } p, \\ 0 & \text{with probability } \frac{1-p}{2}, \\ 1 & \text{with probability } \frac{1-p}{2}. \end{cases}$$

Why are these random restrictions useful? Note that if we hit a k-variable and with a random restriction $\boldsymbol{\rho} \sim \mathcal{R}_p$, then the expected number of surviving variables is $p \cdot k$ (where the survival probability p is thought of as being closer to 0 than it is to 1). But in fact something much stronger happens; with high probability,

$$\Pr[\text{doesn't become the constant 0}] = \left(p + \frac{1-p}{2}\right)^k = \left(\frac{1+p}{2}\right)^k,$$

which is exponentially small in k!

Now let's take a brief detour to discuss the relationship between DNFs, CNFs, and decision trees.

Claim 3.8. *Suppose that F is computed by a depth-d decision tree T. Then F is computed by a width-d CNF and also computed by a width-d DNF.*

Proof. We construct the DNFs and CNFs from the decision tree by constructing the terms and clauses out of the paths in the decision tree which reach the 0s and 1s of the functions, respectively. We will show how to construct width-d DNFs; the construction for CNFs is similar. Each term in the width-d DNF will correspond to a 1-path in the decision tree, and so the width-d DNF will be true if and only if a 1-path can be followed in the decision tree. Let T be a depth-d decision tree computing F, and let P be any path from the root of T to a leaf corresponding to a 1 of F. Let $x_1, \ldots, x_\tau$ be the literals appearing on the path P, where the literal x_i is negated if we move left at the vertex and positive otherwise. For each such path P, we add a term $x_1 \wedge x_2 \wedge \ldots \wedge x_\tau$ to the DNF. If an input x is accepted by the decision tree, then the term corresponding to the accepting path in the tree will also be accepted by the DNF, and the converse direction holds as well. Obviously, for each path P in the decision tree, $\tau \leqslant d$, and so the resulting DNF has at most d literals per term.

The construction for the CNF is entirely analogous, except that we instead consider the paths to the 0s of the decision tree, and add a clause with the negations of the literals followed on the paths. Intuitively, this width-d CNF is true if and only if none of the 0 paths are followed in the decision tree. □

We now state Håstad's switching lemma:

Lemma 3.9 (Håstad's Switching Lemma, [Has86])**.** *Let $f(x_1, \ldots, x_n)$ be computed by a width-w DNF (or CNF). Let the decision tree depth of F be denoted by* DT-depth(f)*. Then for any $t \geqslant 1$, and for $0 < p < 1$,*

$$\Pr_{\boldsymbol{\rho} \sim \mathcal{R}_p} \left[\text{DT-depth}(f\restriction_{\boldsymbol{\rho}}) \geqslant t \right] \leqslant (7 \cdot p \cdot w)^t$$

Let's give a few immediate remarks:

- There is no dependence on the number of variables n or the size of f in the bound, and so the lemma is a very strong statement.
- The lemma also holds for CNFs, and the proof is essentially the same.
- Note that if $p \geqslant 1/7w$, then the statement essentially says nothing; all probabilities are bounded above by 1. If $p < 1/7w$, then the probability of a bad restriction, one that does not signficantly reduce the decision tree depth, is small. For example with $p = 1/14w$, the probability of a bad restriction is at most $1/2^t$.

Next time, we will

- Use Håstad's switching lemma to get the above $2^{\Omega\left(n^{1/(d-1)}\right)}$ lower bound for PARITY_n.
- Give a proof of a baby switching lemma.
- Give a proof of the actual switching lemma.

§4 Lecture 04—06th February, 2024

Last time. In the previous class, we

- Said a little more about formulas; in particular we gave the KRW conjecture and showed that the depth of a boolean formula goes like $\text{depth}(f) = \Theta(\log \text{size}(f))$ for any Boolean formula f.
- Talked a bit about full-basis Boolean formulas, and gave a state-of-the-art $\tilde{\Omega}(n^2)$ lower bound for full-basis formulas via the addressing function.
- Started the unit on constant depth circuits by giving a $2^{O(n^{1/(d-1)})}$-size upper bound for depth-d circuits for the PARITY_n function (to be proved in the homework), and developing some intuition for the proof of the switching lemma (by Håstad) which we stated.

Today. We will:

- Use Håstad's switching lemma to prove a $2^{\Omega(n^{1/(d-1)})}$ lower bound for depth-d circuits for PARITY_n.
- Proof of the baby switching lemma.
- Proof of the strong form of Håstad's switching lemma.
- Provide average-case lower-bounds for AC^0 circuits for PARITY_n.

The readings for today are Sections 6.1, 12.1, and 12.2 of [J+12] and Sections 3.3 and 3.4 of [BS90].

---

§4.1 Worst-case lower bound for PARITY_n in AC^0_d via Håstad's switching lemma

Recall the statement of Håstad's switching lemma:

Lemma 4.1 (Håstad's Switching Lemma, [Has86])**.** *Let $f(x_1, \ldots, x_n)$ be computed by a width-w DNF (or CNF). Let the decision tree depth of F be denoted by* DT-depth(f)*. Then for any $t \geqslant 1$, and for $0 < p < 1$,*

$$\Pr_{\boldsymbol{\rho} \sim \mathcal{R}_p} \left[\text{DT-depth}(f\restriction_{\boldsymbol{\rho}}) \geqslant t \right] \leqslant (7 \cdot p \cdot w)^t$$

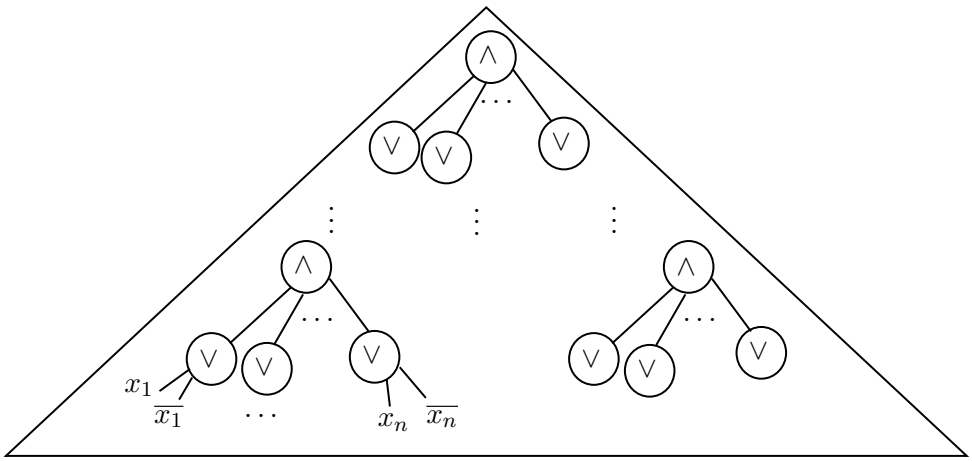
Essentially it tells us that if we impose a suitable random restriction (where each variable survives with probability $p \ll 1/14w$) on a width-w DNF (or CNF) f, then the DNF (or CNF) f becomes a function of a few variables, and simplifies into a decision tree with high probability. Also remember that given a decision tree, we can view it as being a CNF. So this is a "switching lemma" in the sense that it switches between DNF (or CNF) and decision tree representations of a function.

This flexibility is what makes the switching lemma so powerful. We will use it to prove a lower bound for PARITY_n in AC^0_d.

Theorem 4.1. *For any $d \geqslant 2$, any depth-d circuit for PARITY_n has size at least $2^{\Omega(n^{1/(d-1)})}$.*

Proof. We will use the Håstad switching lemma to perform depth reduction on the circuit. Let C be a depth-d circuit of size M for PARITY_n. We'll argue that $M \geqslant 2^{\Omega(n^{1/(d-1)})}$.

Let C be a circuit like AND—OR—…—AND—OR with n inputs.



Stage 0 of the argument proceeds as follows. We will first use a initial restriction to "trim" the circuit with the purpose of reducing the bottom fan-in. Take $p = 1/100$, and hit C with a restriction $\boldsymbol{\rho} \sim \mathcal{R}_{1/100}$, so that with probability $1/100$, the input variable is set to $*$ with probability $1/100$ and to 0 or 1 with probability $99/200$ each. Now if some bottom level gate (an OR) has fanin $\geqslant 10 \log M$, the probability that that gate is fixed to 1 and vanishes is

$$\geqslant 1 - \left(\frac{101}{200}\right)^{10 \log M} \gg 1 - \frac{1}{M^5},$$

and by a union bound over all $\leqslant M$ bottom-level gates, we get that with probability $> 1/2$, this ρ kills all the gates of fanin $\geqslant 10 \log M$. Also, with probability $> 1/2$, at least $n/200$ variables survive the restriction $\boldsymbol{\rho}$. Therefore, there is some restriction $\boldsymbol{\rho}$ so that both these events happen with high probability—fix that $\boldsymbol{\rho}$. Call C_0 the circuit $C\restriction_{\boldsymbol{\rho}}$. Then:

- C_0 computes PARITY_n or its negation on $\geqslant n/200$ variables,
- C_0 has bottom fanin $\leqslant 10 \log M$ and depth d.

Stage 1 applies the switching lemma to each of the depth-2 subcircuits of the bottom section of the circuit. Hit C_0 with a restriction $\boldsymbol{\rho} \sim \mathcal{R}_{1/(100 \log M)}$. Each depth-2 subcircuit at the bottom of C_0 is a $(10 \log M)$-width CNF, and we can apply the switching lemma to each of these subcircuits. Take $t = 10 \log M$, and by the Håstad switching lemma the depth-2 circuit collapses to an $\leqslant 10 \log M$-width decision tree except with failure probability

$$(7pw)^{10 \log M} = \left(\frac{7}{10}\right)^{10 \log M} \leqslant \frac{1}{M^5}.$$

By a union bound over all $\leqslant M$ depth-2 subcircuits, we get that with probability $> 1/2$, all of them collapse to a depth-$(10 \log M)$ decision tree, which can be viewed as a DNF of width $10 \log M$. (Effectively, we have reduced the depth of the circuit by 1 via this stronger restriction.) Furthermore, with probability $> 1/2$, the number of surviving variables under this restriction is $\geqslant (n/200) \cdot (1/(200 \log M))$. Therefore, there is some restriction $\boldsymbol{\rho}$ so that both these events happen with high probability—fix that $\boldsymbol{\rho}$. Call C_1 the circuit $C_0\restriction_{\boldsymbol{\rho}}$. Then:

- C_1 computes PARITY_n or its negation on $\geqslant (n/200) \cdot (1/(200 \log M))$ variables,
- C_1 is a depth-$(d-1)$ circuit with $\leqslant M$ gates at distance 2 from the bottom.

We repeat this procedure to get the following series of reductions: Observe now that C_{d-2} is a depth-2

Current circuit	Depth	Bottom fanin	Number of variables in play
C	d	n	n
C_0	d	$10 \log M$	$n/200$
C_1	$d-1$	$10 \log M$	$n/\left(200 \cdot (200 \log M)\right)$
C_2	$d-2$	$10 \log M$	$n/\left(200 \cdot (200 \log M)^2\right)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
C_{d-2}	2	$10 \log M$	$n/\left(200 \cdot (200 \log M)^{d-2}\right)$

circuit with $\leqslant M$ gates, computes PARITY_n or its negation on $\geqslant (n/200) \cdot \left(1/(200 \log M)^{d-2}\right)$ variables, and is a $10 \log M$-width CNF. So we must have size M satisfying

$$\frac{n}{200 \cdot (200 \log M)^{d-2}} \leqslant 10 \log M \implies M \geqslant 2^{\Omega_d\left(n^{1/(d-1)}\right)}$$

for the circuit C. □

§4.2 A baby switching lemma

Before we give our proof, we first state and prove a baby switching lemma; this will be useful for understanding the proof of Håstad's switching lemma. Here's its statement:

Lemma 4.2 (Baby switching lemma)**.** *Let $f(x_1, \ldots, x_n)$ be computed by a width-w DNF (or CNF). Let the decision tree depth of F be denoted by* DT-depth(f)*. Then for any $t \geqslant 1$, and for $0 < p < 1$,*

$$\Pr_{\boldsymbol{\rho} \sim \mathcal{R}_p} \left[\text{DT-depth}(f\restriction_{\boldsymbol{\rho}}) \geqslant t \right] \leqslant (40 \cdot p \cdot w \cdot 2^w)^t$$

We will spend this section of the class proving this lemma. Before we delve into the prove, it's nice to note the following distinction between the proof of the baby switching lemma and the switching lemma due to Håstad:

1. When we prove Håstad's switching lemma, we first restrict f by $\boldsymbol{\rho}$, and then we analyse the decision tree for $f\restriction_{\boldsymbol{\rho}}$, perhaps by using the fact that the decision tree depth of f is small to argue that the decision tree depth of $f\restriction_{\boldsymbol{\rho}}$ is small.
2. When we prove the baby switching lemma, we first construct the decision tree T for $\boldsymbol{\rho}$, and then take advantage of its nice structure to analyse the decision tree $T\restriction_{\boldsymbol{\rho}}$ under the restriction $\boldsymbol{\rho}$.

Now let's understand what it means for a decsion tree to be hit by a random restriction $\boldsymbol{\rho} \sim \mathcal{R}_p$. Consider a decision tree like

$$T = \begin{array}{ccc} & x_i & \\ \swarrow & & \searrow \\ T_0 & & T_1 \end{array},$$

where T_0 and T_1 are the left and right subtrees of the root of T. When we hit T with a random restriction $\boldsymbol{\rho} \sim \mathcal{R}_p$, there are three possibilities: (1) $T_0\restriction_{\boldsymbol{\rho}}$ if $x_i \leftarrow 0$, (2) $T_1\restriction_{\boldsymbol{\rho}}$ if $x_i \leftarrow 1$, and (3) the tree

$$T = \begin{array}{ccc} & x_i & \\ \swarrow & & \searrow \\ T_0\restriction_{\boldsymbol{\rho}} & & T_1\restriction_{\boldsymbol{\rho}} \end{array},$$

if $x_i \leftarrow *$. This observation motivates the following key definition:

Definition 4.3 (w-clipped decision tree)**.** *A decision tree T is* w-clipped *if every node in T has at least one leaf of distance at most w below it.*

This definition is relevant because of the following lemma:

Lemma 4.4. *Any width-w DNF (or CNF) F is equivalent to (i.e. is computed by) a w-clipped decision tree.*

Proof. Construct a decision tree T by going through F term by term. Read variables in the current term one by one, and move on to the next term when the current term is falsified. Put a 1-leaf where the current term is satisfied, and a 0-leaf if it went through all the terms and none of them were satisfied. By construction, at any node in the tree, there is a 1-leaf of distance at most w from it if and only if the corresponding term is satisfied. Therefore, T is a w-clipped decision tree. □

Given all of the above, to prove Lemma 4.2, we only need prove a switching lemma for w-clipped decision trees:

Lemma 4.5 (Switching Lemma for w-clipped decision trees)**.** *Let T be a w-clipped decision tree of depth d. Then for any $t \geqslant 1$, and for $0 < p < 1$,*

$$\Pr_{\boldsymbol{\rho} \sim \mathcal{R}_p} \left[\text{DT-depth}(T\restriction_{\boldsymbol{\rho}}) \geqslant t \right] \leqslant (40 \cdot p \cdot w \cdot 2^w)^t$$

Now define $\mathsf{Branches}(T)$ as the set of all root-to-leaf paths/branches in T. It is entirely natural to consider a random walk down the decision tree obtained by tossing coins to obtain a natural distribution over the branches of the tree.

Definition 4.6. *Given a decision tree T, let $\mathcal{W}(T)$ be a probability distribution over $\mathsf{Branches}(T)$ given by random walks on T; each branch $\pi \in \mathsf{Branches}(T)$ is chosen with probability $2^{-|\pi|}$, where $|\pi|$ is the length of the branch π. This is equivalent to the uniform distribution on $\mathsf{Branches}(T)$.*

Lemma 4.7. *Let T be a proper decision tree over variables $x_1, \ldots, x_n$ (so no variable occurs twice on any root-to-leaf path). Then the following two distributions $\mathcal{D}_1, \mathcal{D}_2$ over branches are equivalent (recall that a branch is a sequence $\langle \pi_1, \ldots, \pi_n \rangle$ where each π_i is a pair of the form (x_i, b_i), and $b_i \in \{0, 1\}$):*

- *$\mathcal{D}_1(T)$: Draw $\boldsymbol{\rho} \sim \mathcal{R}_p$ and consider $T\restriction_{\boldsymbol{\rho}}$, the subtree of T that is obtained by outputting a branch $\boldsymbol{\sigma} \sim \mathcal{W}(T\restriction_{\boldsymbol{\rho}})$, i.e. $\boldsymbol{\sigma}$ is a branch obtained by doing a random walk down from the root of $T\restriction_{\boldsymbol{\rho}}$. Output $\boldsymbol{\sigma}$.*
- *$\mathcal{D}_2(T)$: Draw $\boldsymbol{\pi} = \langle \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_n \rangle \sim \mathcal{W}(T)$. Output the sub-list of $\boldsymbol{\pi}$ obtained by going through $\boldsymbol{\pi}$ and independently including each element $\boldsymbol{\pi}_i$ with probability p. This is exactly the distribution $\mathsf{Binomial}(|\boldsymbol{\pi}|, p)$.*

Proof. Exercise. □

The following two lemmas will be useful for us as we attempt to prove Lemma 4.5:

Lemma 4.8. *Let $\boldsymbol{X} \in \{w, w+1, \ldots\}$ be a random variable corresponding to the first time that w consecutive heads come up in a sequence of i.i.d. fair coin flips. Prove that $\mathbb{E}\left[\boldsymbol{X}^t\right] \leqslant (7wt2^w)^t$.*

Proof. Exercise. □

Lemma 4.9. *For a w-clipped decision tree T,*

$$\mathbb{E}_{\boldsymbol{\pi} \sim \mathcal{W}(T)} \left[\binom{|\boldsymbol{\pi}|}{t} \right] \leqslant (20w2^w)^t .$$

Proof. Let $\boldsymbol{X}$ be the random variable $\boldsymbol{X} \in \{w, w+1, \ldots\}$ corresponding to the first time that w consecutive heads come up in a sequence of i.i.d. fair coin flips, aligning with taking the correct or incorrect sequence of branches. For any w-clipped decision tree T, $\boldsymbol{X}$ stochastically dominates

$\boldsymbol{\pi} \sim \mathcal{W}(T)$; that is, for all $\alpha \in \mathbb{R}$, $\Pr[X \geqslant \alpha] \geqslant \Pr[|\boldsymbol{\pi}| \geqslant \alpha]$. Therefore for any monotone (nondecreasing) function $g\colon \mathbb{N} \to \mathbb{R}$, $\mathbb{E}[g(\boldsymbol{X})] \geqslant \mathbb{E}[g(|\boldsymbol{\pi}|)]$. Therefore it suffices to show that

$$\mathbb{E}\left[\binom{\boldsymbol{X}}{t}\right] \leqslant (20w2^w)^t.$$

It is given as a homework exercise to show that $\mathbb{E}\left[\boldsymbol{X}^t\right] \leqslant (7wt2^w)^t$. Given this fact, we have that

$$\mathbb{E}\left[\binom{\boldsymbol{X}}{t}\right] \leqslant \mathbb{E}\left[\left(\frac{e\boldsymbol{X}}{t}\right)^t\right] \leqslant \left(\frac{e}{t}\right)^t \cdot \mathbb{E}\left[\boldsymbol{X}^t\right] \leqslant \left(\frac{e}{t}\right)^t \cdot (7wt2^w)^t \leqslant (20w2^w)^t,$$

where the last inequality follows from the fact that $7e \leqslant 20$. □

The intuition for the above lemma is as follows. Since the decision tree is w-clipped, at each node there is a leaf node below it at distance $\leqslant s$. If we interpret the randomly sampled path $\boldsymbol{\pi}$ as a sequence of length-w blocks, then for our path to be multiple blocks long, it must have (at least) made it through each block independently.

Now we are ready to prove the baby switching lemma for w-clipped decision trees.

Proof of Lemma 4.5. Notice that for any restriction $\boldsymbol{\rho}$,

$$\operatorname{depth}\left(T\restriction_{\boldsymbol{\rho}}\right) \geqslant t \implies \Pr_{\boldsymbol{\sigma}\sim\mathcal{W}\left(T\restriction_{\boldsymbol{\rho}}\right)}[|\boldsymbol{\sigma}| \geqslant t] \geqslant 2^{-t},$$

by the definition of the distribution $\mathcal{W}$. Therefore, we have that

$$\Pr_{\boldsymbol{\rho}\sim\mathcal{R}_p}\left[\operatorname{depth}\left(T\restriction_{\boldsymbol{\rho}}\right) \geqslant t\right] \leqslant \Pr_{\boldsymbol{\rho}\sim\mathcal{R}_p}\left[\Pr_{\boldsymbol{\sigma}\sim\mathcal{W}\left(T\restriction_{\boldsymbol{\rho}}\right)}[|\boldsymbol{\sigma}| \geqslant t] \geqslant 2^{-t}\right].$$

Recall now that Markov's inequality states that for any nonnegative random variable $\boldsymbol{X}$ and any $a > 0$, $\Pr[\boldsymbol{X} \geqslant 2^{-a}] \leqslant \mathbb{E}[\boldsymbol{X}] \cdot 2^a$. Therefore, we have that

$$\Pr_{\boldsymbol{\rho}\sim\mathcal{R}_p}\left[\operatorname{depth}\left(T\restriction_{\boldsymbol{\rho}}\right) \geqslant t\right] \leqslant \Pr_{\boldsymbol{\rho}\sim\mathcal{R}_p}\left[\Pr_{\boldsymbol{\sigma}\sim\mathcal{W}\left(T\restriction_{\boldsymbol{\rho}}\right)}[|\boldsymbol{\sigma}| \geqslant t] \geqslant 2^{-t}\right] \leqslant 2^t \cdot \mathbb{E}_{\boldsymbol{\rho}\sim\mathcal{R}_p}\left[\Pr_{\boldsymbol{\sigma}\sim\mathcal{W}\left(T\restriction_{\boldsymbol{\rho}}\right)}[|\boldsymbol{\sigma}| \geqslant t]\right].$$

By Lemma 4.7, we have that the distribution $\mathcal{D}_1(T\restriction_{\boldsymbol{\rho}})$ is equivalent to the distribution $\mathcal{D}_2(T\restriction_{\boldsymbol{\rho}})$, and so

$$\begin{aligned}
\Pr_{\boldsymbol{\rho}\sim\mathcal{R}_p}\left[\operatorname{depth}\left(T\restriction_{\boldsymbol{\rho}}\right) \geqslant t\right] &\leqslant 2^t \cdot \mathbb{E}_{\boldsymbol{\rho}\sim\mathcal{R}_p}\left[\Pr_{\boldsymbol{\sigma}\sim\mathcal{W}\left(T\restriction_{\boldsymbol{\rho}}\right)}[|\boldsymbol{\sigma}| \geqslant t]\right] && \text{by Markov,}\\
&= 2^t \cdot \mathbb{E}_{\boldsymbol{\pi}\sim\mathcal{W}(T)}\left[\Pr_{\boldsymbol{Y}\sim\mathsf{Binom}(|\boldsymbol{\pi}|,p)}[\boldsymbol{Y} \geqslant t]\right] && \text{by Lemma 4.7,}\\
&\leqslant 2^t \cdot \mathbb{E}_{\boldsymbol{\pi}\sim\mathcal{W}(T)}\left[p^t \cdot \binom{|\boldsymbol{\pi}|}{t}\right]\\
&= (2p)^t \cdot \mathbb{E}_{\boldsymbol{\pi}\sim\mathcal{W}(T)}\left[\binom{|\boldsymbol{\pi}|}{t}\right] && \text{by linearity of expectation,}\\
&\leqslant (2p)^t \cdot (20w2^w)^t && \text{by Lemma 4.9,}\\
&= (40pw2^w)^t.
\end{aligned}$$

This completes the proof. □

We have now proved the baby switching lemma for w-clipped decision trees. Since any width-w DNF (or CNF) is equivalent to a w-clipped decision tree, we get a proof of the baby switching lemma for width-w DNFs and CNFs for free.

We will now use ideas from this lemma to prove Håstad's switching lemma.

§4.3 Proof of Håstad's switching lemma

We will prove the following version of the switching lemma due to Håstad:

Lemma 4.10. *Let $f(x_1, \ldots, x_n)$ be computed by a width-w DNF (or CNF). Let the decision tree depth of F be denoted by* DT-depth(f). *Then for any $t \geqslant 1$, and for $0 < p < 1$,*

$$\Pr_{\boldsymbol{\rho} \sim \mathcal{R}_p} [\text{DT-depth}(f\restriction_{\boldsymbol{\rho}}) \geqslant t] \leqslant (10 \cdot p \cdot w)^t$$

Here we're thinking of p as a small number, really $\leqslant 1/10w$, so that we do not get trapped in trivialities. Furthermore, we consider the assignation $x_i \leftarrow *$ by the restriction $\boldsymbol{\rho}$ as being unlikely, because each individual variable probably gets fixed to 0 or 1.

Now consider the width-w DNF $F = T_1 \vee T_2 \vee \ldots$, where the terms T_i are arranged in a fixed order. What happens to one term, say $T_i = x_1 \wedge \overline{x_3} \wedge x_4$ under a restriction $\boldsymbol{\rho} \sim \mathcal{R}_p$?

1. Most likely, some literal could get set to 0, and then $T_i\restriction_{\boldsymbol{\rho}} \equiv 0$, i.e. the term T_i vanishes from $F\restriction_{\boldsymbol{\rho}}$. If this happens to every T_i, then $F\restriction_{\boldsymbol{\rho}} \equiv 0$.
2. Also, we could have that $T_i\restriction_{\boldsymbol{\rho}} \equiv 1$, i.e. the term T_i survives the restriction $\boldsymbol{\rho}$. If this happens to at least one T_i, then $F\restriction_{\boldsymbol{\rho}} \equiv 1$.
3. Otherwise, perhaps some literals in T_i gets fixed to $*$, and some other literals gets fixed to 1. Thus T_i potentially "shrinks", but survives.

So overall there are three possibilities for the fate of F when hit with the random restriction $\boldsymbol{\rho} \sim \mathcal{R}_p$. It either gets fixed to 0, fixed to 1, or some T_i vanish to 0 while others shrink.

Example 4.11. Consider the following example. Apply the restriction

$$\boldsymbol{\rho}: \quad x_1 \leftarrow *, \quad x_2 \leftarrow 1, \quad x_3 \leftarrow *, \quad x_4 \leftarrow 1, \quad x_5 \leftarrow *, \quad x_6 \leftarrow 0, \quad x_7 \leftarrow *$$

to the width-3 DNF

$$F = (x_1 \wedge \overline{x_2} \wedge x_4) \vee (x_2 \wedge x_5 \wedge \overline{x_6}) \vee (x_3 \wedge \overline{x_5} \wedge x_7)\,.$$

Then

$$F = \overbrace{\begin{pmatrix} x_1 & \wedge & \overline{x_2} & \wedge & x_4 \\ * & & \overline{1} & & 1 \end{pmatrix}}^{T_1} \vee \overbrace{\begin{pmatrix} x_2 & \wedge & x_5 & \wedge & \overline{x_6} \\ 1 & & * & & \overline{0} \end{pmatrix}}^{T_2} \vee \overbrace{\begin{pmatrix} x_3 & \wedge & \overline{x_5} & \wedge & x_7 \\ * & & * & & * \end{pmatrix}}^{T_3}.$$

Now $T_1\restriction_{\boldsymbol{\rho}} \equiv 0$, $T_2\restriction_{\boldsymbol{\rho}} \equiv x_5$, and $T_3\restriction_{\boldsymbol{\rho}} \equiv x_3 \wedge \overline{x_5} \wedge x_7$. Therefore, $F\restriction_{\boldsymbol{\rho}} \equiv x_5 \vee (x_3 \wedge \overline{x_5} \wedge x_7)$. Qualitatively speaking, the restriction $\boldsymbol{\rho}$ has "shrunk" the DNF F to a smaller DNF $F\restriction_{\boldsymbol{\rho}}$.

Our goal is then to show that with high probability over a random restriction $\boldsymbol{\rho} \sim \mathcal{R}_p$, some decision tree computing the restricted function $F\restriction_{\boldsymbol{\rho}}$ has small depth (i.e. is shallow). We will do this by arguing that with high probability over $\boldsymbol{\rho} \sim \mathcal{R}_p$, there is a particular way of writing a decision tree for $F\restriction_{\boldsymbol{\rho}}$ such that it is shallow.

Definition 4.12 (Canonical decision tree for F after $\boldsymbol{\rho}$)**.** *Let $F = T_1 \vee T_2 \vee \dots$ be a width-w DNF (or CNF), and let $\boldsymbol{\rho}$ be a restriction. The* canonical decision tree $\mathsf{CDT}(F, \boldsymbol{\rho})$ *for F after $\boldsymbol{\rho}$ is the decision tree obtained by the following recursive process:*

- *If $F\restriction_{\boldsymbol{\rho}}$ is killed to the constant 0 or the constant 1, then $\mathsf{CDT}(F, \boldsymbol{\rho})$ is the decision tree of depth 0 that outputs the constant 0 or the constant 1.*
- *Otherwise, $F\restriction_{\boldsymbol{\rho}}$ wasn't killed by the restriction, and some term T_i survives the restriction. Find the first such term. Then $\mathsf{CDT}(F, \boldsymbol{\rho})$ obliviously/exhaustively queries the block of all the surviving variables in that term T_i.*
- *Recurse at each leaf: for each leaf, do the above two steps under the augumented restriction corresponding to the leaf.*

Note that crucially, when the canonical decision tree $\mathsf{CDT}(F, \boldsymbol{\rho})$ queries a block of variables, there is a unique assignment/path that satisfies all the literals in that restricted term corresponding to the block, and we get a 1-leaf in the canonical decision tree there.

Example 4.13. Recall the example above, where we applied the restriction

$$\boldsymbol{\rho}: \quad x_1 \leftarrow *, \quad x_2 \leftarrow 1, \quad x_3 \leftarrow *, \quad x_4 \leftarrow 1, \quad x_5 \leftarrow *, \quad x_6 \leftarrow 0, \quad x_7 \leftarrow *$$

to the width-3 DNF

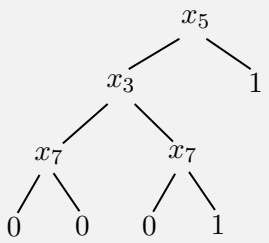
$$F = (x_1 \wedge \overline{x_2} \wedge x_4) \vee (x_2 \wedge x_5 \wedge \overline{x_6}) \vee (x_3 \wedge \overline{x_5} \wedge x_7)$$

to get the restricted DNF

$$F\restriction_{\boldsymbol{\rho}} \equiv x_5 \vee (x_3 \wedge \overline{x_5} \wedge x_7)\,.$$

The canonical decision tree $\mathsf{CDT}(F, \boldsymbol{\rho})$ for F after $\boldsymbol{\rho}$ is then

$\mathsf{CDT}(F, \boldsymbol{\rho}) =$



The first block above is root x_5 (as it is the first surviving term), and the second block above is the left subtree.

Now define bad to be the set of restrictions $\boldsymbol{\rho}$ such that $\mathsf{CDT}(F, \boldsymbol{\rho})$ has depth $\geqslant t$. We will show that $\Pr_{\boldsymbol{\rho} \sim \mathcal{R}_p}[\boldsymbol{\rho} \in \mathsf{bad}] \leqslant (10 \cdot p \cdot w)^t$. If we can accomplish this, then we will have proved Lemma 4.10.

To prove this, we start by fixing a restriction $\boldsymbol{\rho} \in \mathsf{bad}$, so that $\mathsf{CDT}(F, \boldsymbol{\rho})$ has depth $\geqslant t$. Let P be the leftmost path of depth $\geqslant t$ in $\mathsf{CDT}(F, \boldsymbol{\rho})$. We assume the depth of P is exactly t (if it is greater

than t, then we can just truncate it to depth t). Therefore, the combined object $\boldsymbol{\rho} \cup P$ is fixes t more variables than $\boldsymbol{\rho}$ does. Call this restriction $\mathsf{Devil}(\boldsymbol{\rho})$. Note that because of the structure of the canonical decision tree, $\mathsf{Devil}(\boldsymbol{\rho})$ has "segments" based on the blocks of $\mathsf{CDT}(F, \boldsymbol{\rho})$ that were queried along P. Call the ith such block V_i, and let δ_i be the restriction fixing V_i as in $\mathsf{Devil}(\boldsymbol{\rho})$. Then $\mathsf{Devil}(\boldsymbol{\rho})$ is the combined restriction $\boldsymbol{\rho} \cup \delta_1 \cup \delta_2 \cup \ldots \cup \delta_k$, where $k \leqslant t$.

We now introduce the hero of this play. Consider another restriction $\mathsf{Angel}(\boldsymbol{\rho})$, which, like $\mathsf{Devil}(\boldsymbol{\rho})$, fixes t more additional variables beyond $\boldsymbol{\rho}$, and in particular, fixes the same ones V_i, but in such a way so as to reach the 1-leaf of the block V_i. As a consequence, it is not a path in the tree—it is disconnected. Furthermore, $\mathsf{Angel}(\boldsymbol{\rho})$ and $\mathsf{Devil}(\boldsymbol{\rho})$ are disjoint, and so $\mathsf{Angel}(\boldsymbol{\rho})$ disagrees with $\mathsf{Devil}(\boldsymbol{\rho})$ in each block.

The following example should clarify both $\mathsf{Devil}(\boldsymbol{\rho})$ and $\mathsf{Angel}(\boldsymbol{\rho})$. Consider the diagram to the right. (Note that in the diagram the red lines are the actions of the Devil, and the green lines are the actions of the Angel.) Take the restriction

$$\boldsymbol{\rho}: \quad x_1 \leftarrow 1,\ x_2 \leftarrow *,\ x_3 \leftarrow 0,\ x_4 \leftarrow *,\ x_5 \leftarrow *.$$

Then:

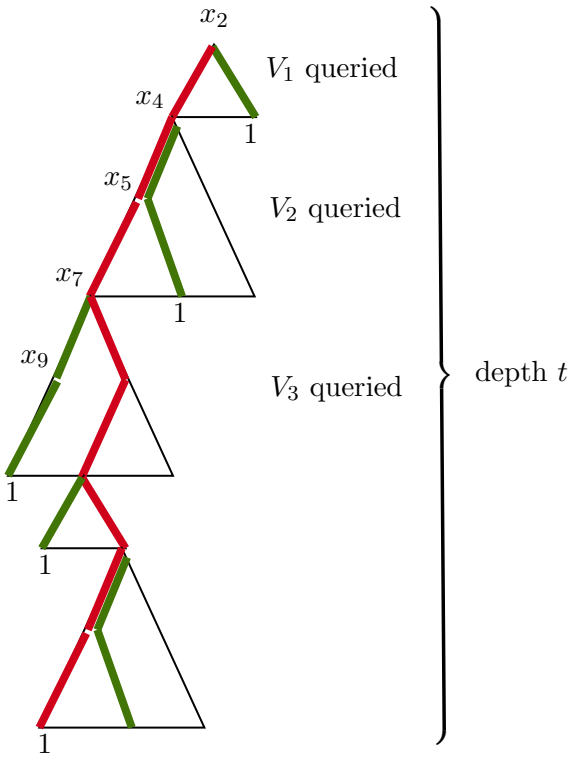
$$V_1 = \text{first block} = \{x_2\},$$
$$\text{and the unkilled term of } F\restriction_{\boldsymbol{\rho}} \text{ is } x_2.$$

Similarly,

$$V_2 = \text{second block} = \{x_4, x_5\},$$
$$\text{and the unkilled term of } F\restriction_{\boldsymbol{\rho}} \text{ is } \overline{x_4} \wedge x_5,$$

and

$$V_3 = \text{third block} = \{x_7, x_9\},$$
$$\text{and the unkilled term of } F\restriction_{\boldsymbol{\rho}} \text{ is } \overline{x_7} \wedge \overline{x_9}.$$



Then after the entire melee,

	x_1	x_2	x_3	x_4	x_5
$\boldsymbol{\rho}$	1	$*$	0	$*$	$*$
$\mathsf{Devil}(\boldsymbol{\rho})$	1	0	0	0	0
$\mathsf{Angel}(\boldsymbol{\rho})$	1	1	0	0	1

Now consider the following question. Given a restriction $\boldsymbol{\rho} \in \mathsf{bad}$, is $\mathsf{Angel}(\boldsymbol{\rho})$ or $\boldsymbol{\rho}$ more likely to be drawn under the distribution $\mathcal{R}_p$? In fact, $\mathsf{Angel}(\boldsymbol{\rho})$ is much more likely to be drawn. This is because $\mathsf{Angel}(\boldsymbol{\rho})$ is a restriction that has more fixed bits, and these fixed bits are much more likely than $*$ to be drawn under the distribution $\mathcal{R}_p$. We will show this more precisely now. Fix any particular $\boldsymbol{\rho} \in \{0, 1, *\}^n$ with k many $*$s. Then it follows that the probability of drawing $\boldsymbol{\rho}$ under

the distribution $\mathcal{R}_p$ is

$$\Pr_{\boldsymbol{\rho}\sim\mathcal{R}_p}[\boldsymbol{\rho}] = p^k \cdot \left(\frac{1-p}{2}\right)^{n-k}.$$

So for any $\boldsymbol{\rho} \in \mathsf{bad}$ with k many $*$s, we have that

$$\begin{aligned}\frac{\Pr_{\boldsymbol{\rho}\sim\mathcal{R}_p}[\boldsymbol{\rho}]}{\Pr_{\boldsymbol{\rho}\sim\mathcal{R}_p}[\mathsf{Angel}(\boldsymbol{\rho})]} &= \frac{p^k \cdot \left(\frac{1-p}{2}\right)^{n-k}}{p^{k-t} \cdot \left(\frac{1-p}{2}\right)^{n-(k-t)}} \\ &= \left(\frac{2p}{1-p}\right)^t \leqslant (2.5p)^t,\end{aligned}$$

since p is small and $\mathsf{Angel}(\boldsymbol{\rho})$ has $k-t$ many $*$s. Therefore, $\mathsf{Angel}(\boldsymbol{\rho})$ is much more likely to be drawn than $\boldsymbol{\rho}$ under the distribution $\mathcal{R}_p$; in particular,

$$\Pr_{\boldsymbol{\rho}\sim\mathcal{R}_p}[\boldsymbol{\rho}] \leqslant (2.5p)^t \cdot \Pr_{\boldsymbol{\rho}\sim\mathcal{R}_p}[\mathsf{Angel}(\boldsymbol{\rho})].$$

The proof of Håstad's switching lemma is all but done; we only need to show this one key fact:

Lemma 4.14. *Any restriction $\boldsymbol{\sigma}$ is $\mathsf{Angel}(\boldsymbol{\rho})$ for $\leqslant (4w)^t$ many bad restrictions $\boldsymbol{\rho}$.*

Now, although we do not prove this today, the proof of the lemma follows right away if we assume it to be true. This is because we can simply sum over all the bad restrictions $\boldsymbol{\rho} \in \mathsf{bad}$ to get

$$\begin{aligned}\Pr_{\boldsymbol{\rho}\sim\mathcal{R}_p}[\boldsymbol{\rho} \in \mathsf{bad}] &= \sum_{\boldsymbol{\rho}\in\mathsf{bad}} \Pr_{\boldsymbol{\rho}\sim\mathcal{R}_p}[\boldsymbol{\rho}] \\ &\leqslant \sum_{\boldsymbol{\rho}\in\mathsf{bad}} (2.5p)^t \cdot \Pr_{\boldsymbol{\rho}\sim\mathcal{R}_p}\left[\underbrace{\mathsf{Angel}(\boldsymbol{\rho})}_{=\boldsymbol{\sigma}}\right] \\ &\leqslant \sum_{\boldsymbol{\rho}\in\mathsf{bad}} (4w)^t \cdot \Pr_{\boldsymbol{\rho}\sim\mathcal{R}_p}[\boldsymbol{\sigma}] \\ &\leqslant (4w)^t \cdot (2.5p)^t \cdot \left(\sum_{\boldsymbol{\sigma}} \Pr_{\boldsymbol{\rho}\sim\mathcal{R}_p}[\boldsymbol{\sigma}]\right) \\ &\leqslant (4w)^t \cdot (2.5p)^t \\ &\leqslant (10 \cdot p \cdot w)^t,\end{aligned}$$

since $\sum_{\boldsymbol{\sigma}} \Pr_{\boldsymbol{\rho}\sim\mathcal{R}_p}[\boldsymbol{\sigma}] = 1$ (i.e. it is a sum over all the elements of the domain of $\boldsymbol{\sigma}$).

This completes the proof of Håstad's switching lemma—a beautiful result.

Next time, we will:

- Prove Lemma 4.14 via a nice sufficient-encoding argument.
- Give average case AC^0 lower bounds.
- Give (quantitatively strong) average case lower bounds for DNFs and CNFs via random projections.
- Introduce $\mathbb{F}_2$-polynomials.

§5 Lecture 05—13th February, 2024

Last time. In the previous class, we

- Used Håstad's switching lemma to prove a $2^{\Omega(n^{1/(d-1)})}$ lower bound for depth-d circuits for PARITY_n.
- Proved a weak form of Håstad's switching lemma.
- Started the proof of the strong form of Håstad's switching lemma.

Today. We will:

- Finish the proof of the strong form of Håstad's switching lemma by proving the following fact: any restriction $\boldsymbol{\sigma}$ is $\mathsf{Angel}(\boldsymbol{\rho})$ for $\leqslant (4w)^t$ many bad restrictions $\boldsymbol{\rho}$.
- Introduce average-case lower bounds for AC^0 circuits for PARITY_n, which will be a small extension of the worst-case lower bounds we already derived.
- Introduce average-case lower bounds for depth-2 circuits via the argument by O'Donnell and Wimmer: any CNF that agrees with some explicit function f on 90% of the inputs must have $2^{\Omega\left(\frac{n}{\log n}\right)}$ clauses.
- Start $\mathbb{F}_2$-polynomials and start an average-case lower bound for them.

§5.1 Finishing the proof of Håstad's switching lemma

Recall that we were trying to prove the following fact to complete the proof of the strong form of Håstad's switching lemma by showing that any restriction $\boldsymbol{\sigma}$ cannot be $\mathsf{Angel}(\boldsymbol{\rho})$ for too many bad restrictions $\boldsymbol{\rho}$.

Lemma 5.1. *Any restriction $\boldsymbol{\sigma}$ is $\mathsf{Angel}(\boldsymbol{\rho})$ for $\leqslant (4w)^t$ many bad restrictions $\boldsymbol{\rho}$.*

Proof. Fix a restriction $\boldsymbol{\sigma} = \mathsf{Angel}(\boldsymbol{\rho})$ for some bad restriction $\boldsymbol{\rho}$. We will demonstrate how to recover (via a decoding procedure) $\boldsymbol{\rho}$ from $\boldsymbol{\sigma}$ given just a little bit of auxiliary information. (Note that in the entire argument F is known and always has been; there is no secracy about F.) The key idea here is that if we bound the number of possibilities for this auxiliary bit of information, then we have a bound on the number of possible bad restrictions $\boldsymbol{\rho}$ that $\boldsymbol{\sigma}$ can be Angel for.

The auxiliary information here will be the row consisting of 2 rows of t numbers plus a little extra information. In particular, the first row will consist of elements of $\{1, \ldots, w\}^t$ (with w^t many possibilities), the second row will consist of elements of $\{0,1\}^t$ (with 2^t many possibilities), and extra information will be the following: for each of the $t-1$ positions between 2 elements of the first row, put ; or put nothing. This gives us 2^{t-1} possibilities for the second row. So there are $w^t \cdot 2^t \cdot 2^{t-1} \leqslant (4w)^t$ many possibilities here.

Example 5.2. Here's an example of the auxiliary information might look like:

$$\begin{array}{ccccccccccccc} 2 & \_ & 3 & ; & 5 & \_ & 4 & \_ & 5 & ; & \cdots & \_ & 3 \\ 0 & & 0 & & 1 & & 0 & & 1 & & & & 1 \end{array}$$

Now, if there is some deterministic fixed unambiguous way to recover $\boldsymbol{\rho}$ from $\boldsymbol{\sigma}$ and the auxiliary information, then we are done, as there are at most $(4w)^t$ many possibilities for $\boldsymbol{\rho}$ that $\boldsymbol{\sigma}$ can be Angel for. So we need to describe how to decode $\boldsymbol{\rho}$ from $\boldsymbol{\sigma}$ and the auxiliary information. The difficulty here is that we don't know which t fixed bits of $\boldsymbol{\sigma}$ are from $\mathsf{Angel}(\boldsymbol{\rho})$ and which are already fixed in $\boldsymbol{\rho}$. If we know, we could just recover $\boldsymbol{\rho}$ by looking at the bits of $\boldsymbol{\sigma}$ and replacing the fixed bits of $\boldsymbol{\sigma}$ with $*$'s.

We'll proceed by identifying the variables block by block by first finding V_1, then V_2, and so on. Here's how we will do this. Recall that the canonical decision tree $\mathsf{CDT}(F)$ has V_1 surviving variables in the first level of F that is not killed to 0 by $\boldsymbol{\rho}$. Imagine restricting F by $\boldsymbol{\sigma}$. Now $\boldsymbol{\sigma}$ is an extension of $\boldsymbol{\rho}$, and so any term that is killed to 0 by $\boldsymbol{\sigma}$ is also killed to 0 by $\boldsymbol{\rho}$. But the first term in F that is not killed to 0 by $\boldsymbol{\rho}$ is satisfied by $\boldsymbol{\sigma} = \mathsf{Angel}(\boldsymbol{\rho})$, and therefore the first term in F that is not killed to 0 by $\boldsymbol{\rho}$ is where V_1 came from.

Example 5.3. Applying the restriction

x	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
$\boldsymbol{\sigma}$	0	0	1	1	1	0	1	1

to the Boolean function

$$F = \underbrace{(x_1 \wedge \overline{x_2})}_{\substack{\text{killed by } \boldsymbol{\sigma}\text{, so not} \\ \text{where } V_1 \text{ came from}}} \vee \underbrace{(\overline{x_1} \wedge \overline{x_2} \wedge x_8)}_{\substack{\text{satisfied by } \boldsymbol{\sigma}\text{, so} \\ V_1 \text{ came from here!}}} \vee (\overline{x_2} \wedge x_4 \wedge x_5) \vee \cdots,$$

we can narrow things down. The first term in F that is not killed to 0 by $\boldsymbol{\rho}$ is where V_1 came from. In this case, $V_1 = \{x_1\}$. The idea here is very similar to the same idea as before where we used the unkilled terms from the definition of the Angel and the Devil to find the branching paths in the canonical decision tree that eventually lead to the variables in the block.

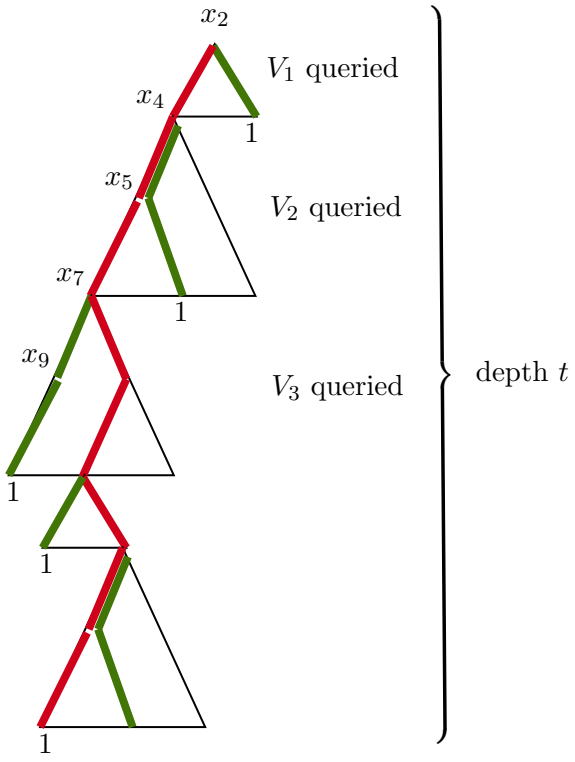
$$\begin{aligned} V_1 = \text{first block} = \{x_1\}, \\ \text{and the unkilled term of } F\restriction_{\boldsymbol{\rho}} \text{ is } x_1. \end{aligned}$$

Recall our analysis from last time for the rest of the argument.

Now here's a new question: which of the variables in the term we just found are the V_1 variables? We can't just look at the bits of $\boldsymbol{\sigma}$ and replace the fixed bits of $\boldsymbol{\sigma}$ with $*$'s, because we don't know which bits of $\boldsymbol{\sigma}$ are fixed by $\boldsymbol{\rho}$ and which are fixed by $\boldsymbol{\sigma}$ itself. But we can use the auxiliary information to figure this out. The auxiliary information tells us which bits of $\boldsymbol{\sigma}$ are fixed by $\boldsymbol{\rho}$ and which are fixed by $\boldsymbol{\sigma}$ itself. So we can use the auxiliary information by reading the width-w term—the elements of the first row in particular—to get information about which of the w variables in the term are $*$'s in $\boldsymbol{\rho}$, and the breakpoint (;) indicates when we're done for a particular term.

So now we have found V_1. We now move on to V_2. We use the second row to learn how to traverse the V_1 block of the canonical decision tree to follow the $\mathsf{Devil}(\boldsymbol{\rho})$ path—that is, the second row tells us how $\mathsf{Devil}(\boldsymbol{\rho})$ would have fixed the variables in the V_1 block.

Now morph $\boldsymbol{\sigma} \rightsquigarrow \boldsymbol{\sigma}'$ by replacing V_1 variables with those bits, and then continue. The first term in F satisfied by $\boldsymbol{\sigma}'$ is where V_2 came from.



Given the canonical decision tree to the left (the first 'triangle' corresponds to V_1, etc), we can continue the process in exactly this way to find V_2, V_3, and so on. The auxiliary information tells us how to traverse the canonical decision tree to find the next block of variables. The first term in F satisfied by $\boldsymbol{\sigma}$ is where V_2 came from.

In the end, we can recover $\boldsymbol{\rho}$ from $\boldsymbol{\sigma}$ and the auxiliary information by the decoding process herein described, replacing all the V_1, V_2, etc variables in $\boldsymbol{\sigma}$ with $*$'s. The auxiliary information tells us which bits of $\boldsymbol{\sigma}$ are fixed by $\boldsymbol{\rho}$ and which are fixed by $\boldsymbol{\sigma}$ itself. So we can use the auxiliary information to decode $\boldsymbol{\rho}$ from $\boldsymbol{\sigma}$.

This completes the proof of the lemma, since we already know that there are at most $(4w)^t$ many possibilities for the auxiliary information by the combinatorial argument we gave earlier in the proof.

Therefore, we have shown that any restriction $\boldsymbol{\sigma}$ is $\mathsf{Angel}(\boldsymbol{\rho})$ for $\leqslant (4w)^t$ many bad restrictions $\boldsymbol{\rho}$. □

§5.2 Average-case lower bounds for AC^0 circuits

We have already seen that by Håstad's switching lemma, PARITY_n requires depth-d circuits of size $2^{\Omega(n^{1/(d-1)})}$, and this is a worst-case lower bound for PARITY_n. We will now use similar techniques—basically reusing earlier work—to show average-case lower bounds for PARITY_n for AC^0 circuits. In essence, we can go over the previous arguments, and note that all our "with probability $\geqslant 1/2$" statements are actually very strong.

Fix a parameter $M = 2^{cn^{1/d}}$, where $c = c_d = 1/100^d$ is a small constant that depends on d. We can verify that for $M = 2^{cn^{1/d}}$, we have that each failure probability $\leqslant 1/M^5$. Now, in the argument in Section 4.1, we were worried about $2d - 2 = O(d)$ many failure events for the switching probability, and therefore, we have an overall probability $\geqslant 1 - O(d)/M^5$, the circuit C_{d-2} is a depth-2 circuit with bottom fanin $\leqslant 10 \log M$, over $n_{d-2} \geqslant n \cdot \left(200\,(200 \log M)^{d-2}\right)^{-1}$ many variables.

Now do one more random restriction, with $p = (100 \log M)^{-1}$ and $t = 10 \log M$. Then it naturally follows that

$$\begin{aligned}\Pr\left[C_{d-2} \text{ doesn't collapse to a depth-}(10 \log M) \text{ decision tree}\right] &\leqslant \left(7 \cdot \frac{1}{100 \log M} \cdot 10 \log M\right)^{10 \log M} \\ &\leqslant \frac{1}{M^5},\end{aligned}$$

and so by the Chernoff bound,

$$\Pr\left[\text{fewer than } \frac{n_{d-2}}{200\log M} \text{ many variables survive}\right] \leqslant \exp\left(-\frac{n}{200\log M}\cdot\frac{1}{2}\right) \leqslant \frac{1}{M^5},$$

from our choice for M.

So overall, with probability $\geqslant 1 - O(d)/M^5$, we have that C_{d-2} collapses to a depth-$(10\log M)$ decision tree with $\geqslant n\cdot\left(c\cdot(\log M)^{d-1}\right)^{-1}$ many variables. We now use the following fact:

Fact 5.4. *Any decision tree of depth d has correlation 0 under the uniform distribution $\mathcal{U}$ with any* PARITY *function on $> d$ variables.*

Proof. Every path in the decision tree is a conjunction of literals which are right half the time and wrong half the time, and so the decision tree has correlation 0 with any PARITY function on $> d$ variables. □

Now we want to think of the random restriction as first picking the fixed variables and deciding how to fix them; one of these choices will be such that there is zero advantage in this situation. So the decision tree only gets it right with probability $O(d)/M^5 = \varepsilon$. Reinterpreting this, we obtain the following average-case lower bound.

Theorem 5.1 (Average-case lower bound for AC^0 circuits for PARITY_n)**.** *Let C be a circuit of size $M = 2^{cn^{1/d}}$ and depth d, with c an absolute constant. Then PARITY_n is ε-hard for C under the uniform distribution $\mathcal{U}$, where $\varepsilon = O(d)/M^5 \leqslant 2^{-cn^{1/d}}$.*

It would be awesome to show that for some explicit function f (not necessarily PARITY_n), we can make the size bound (of the circuits that we get a correlation bound against) bigger, and the correlation bound smaller. But we don't know how to do this. However, Ryan O'Donnell and Karl Wimmer have presented trade-off type results like this for $d = 2$ circuits.

§5.3 O'Donnell-Wimmer average-case lower bound for depth-2 circuits

O'Donnell and Wimmer gave a beautifully simple argument for making the size bound bigger and the correlation bound also bigger in a favourable way, although not for PARITY_n but only depth-2 circuits. Indeed, if $d = 2$ above, we would get a size bound of $\varepsilon \leqslant 2^{n^{1/d}} = 2^{\sqrt{n}}$, and correlation bound of $\varepsilon \leqslant 2^{-\sqrt{n}}$. We will make the correlation bound go all the way to a constant, and the size go to almost 2^n, which is quite impressive.

Consider the following function $\mathrm{DNFTRIBES}_n\colon \{0,1\}^{w2^w} \to \{0,1\}$, which is defined as follows:

$$\mathrm{DNFTRIBES}(x_1,\ldots,x_n) = (x_1\wedge\ldots\wedge x_w)\vee(x_{w+1}\wedge\ldots\wedge x_{2w})\vee\cdots\vee(x_{n-w+1}\wedge\ldots\wedge x_n),$$

where $n = w2^w$ and every variable occurs in exactly one term. There are $n/\log n = 2^w$ many terms, with w variables in each term, and $w \approx \log n - \log\log n$. The following theorem which we will prove says the following: the function $\mathrm{DNFTRIBES}_n$ is hard to approximate by a CNF unless the CNF is enormous—there is some incompatibility between DNF and CNF representations of functions.

Theorem 5.2 (O'Donnell and Wimmer, [OW07])**.** *Any CNF g that agrees with* $\mathrm{DNFTRIBES}_n$ *on* 90% *of all* 2^n *inputs must have* $2^{\Omega\left(\frac{n}{\log n}\right)}$ *many clauses.*

We can see that the correlation bound is constant, and the size bound is almost exponential in n. We'll see that by extending this we can get an average-case lower bound for all depth-2 circuits in the homework. For now we focus on proving this theorem.

The first step of the argument is to convert it into saying something about CNFs of bounded width.

Fact 5.5. *If g is an s-clause CNF, then there is a CNF g' which is ε-close to g, i.e.*

$$\Pr_{\boldsymbol{u}\sim\mathcal{U}_s}[g(\boldsymbol{u}) = g'(\boldsymbol{u})] \geqslant 1-\varepsilon,$$

such that g' has width of every clause $\leqslant \log(s/\varepsilon)$.

Proof. Any clause of length t is falsified with probability $1/2^t$. So removing a clause of width $\geqslant \log(s/\varepsilon)$ changes the function on $\leqslant 1/2^{\log(s/\varepsilon)} = \varepsilon/s$ fraction of inputs. Since g only has s clauses overall, so the total effect from removing all these clauses is $\leqslant s \cdot \varepsilon/s = \varepsilon$. By a union bound over all $\leqslant s$ such clauses removed, we get

$$\Pr_{\boldsymbol{u}\sim\mathcal{U}_s}[g(\boldsymbol{u}) \neq g'(\boldsymbol{u})] \leqslant \varepsilon,$$

and so g' is ε-close to g. □

Given this fact, we know that if we want to prove that any CNF that agrees with $\mathrm{DNFTRIBES}_n$ on 90% of the inputs must have a lot of clauses, it is enough to prove that any CNF that agrees on, say 80% of the inputs, has to be very wide. Let's confirm formally that this is enough. Suppose g is a 0.1-approximator of $\mathrm{DNFTRIBES}_n$ and has s clauses. Then by Fact 5.5, there is a width-$\log(10s)$ CNF g' which is 0.1-close to g. So g' is a 0.2-approximator of $\mathrm{DNFTRIBES}_n$, and so by our reduced statement (which we are showing implies 5.2), we have that

$$\log(10s) \geqslant \Omega\left(\frac{n}{\log n}\right),$$

and so the implication is indeed correct.

We now want to prove the following lemma, which would imply Theorem 5.2.

Lemma 5.6. *Any CNF g that agrees with* $\mathrm{DNFTRIBES}_n$ *on* 80% *of all* 2^n *inputs must have width*

$$\geqslant \frac{1}{4}\cdot 2^w = \Omega\left(\frac{n}{\log n}\right).$$

Notice that we have no hope that random restrictions or the switching lemma will help us here, because the function $\mathrm{DNFTRIBES}_n$ is a narrow DNF; every term is width-w, and we're trying to argue that much wider CNFs can't do well on it. So if we hit g' with a random restriction and simplify it, we will entirely destroy $\mathrm{DNFTRIBES}_n$, which is not ideal for us, as in some sense the hard function $\mathrm{DNFTRIBES}_n$ is much easier than its computer, g'. So we need a way to "keep $\mathrm{DNFTRIBES}_n$ complex" while "making g' simple," and we will do that by an extension of random restrictions called random projections.

§5.3.1 Random projections

A random projection is essentially a "smarter" version of a random restriction:

Definition 5.7. *A projection $\boldsymbol{\rho}$ is a mapping*

$$\{x_1, x_2, \ldots, x_n\} \mapsto \{0, 1, y_1, y_2, \ldots, y_t\}$$

that sends the variables x_i to either 0, 1, or one of a collection of variables y_j.

In a sense we may think of the projection as a mapping that fixes variables and identifies groups of variables.

Example 5.8. Here's an example of a projection:

x_1	x_2	x_3	x_4	x_5	x_6
1	0	y_1	y_1	y_2	y_2

Clearly x_1 is fixed to 1, x_2 is fixed to 0, and x_3 and x_4 are identified with each other, and so are x_5 and x_6.

Just like before, we write $f\restriction_{\boldsymbol{\rho}}$ to denote the function f after applying the projection $\boldsymbol{\rho}$ to it. The advantage of the projection is that it lets us "carefully preserve structure" in the target function, which in our argument will be the read-once DNF DNFTRIBES_n, so that it "survives."

The key to the argument is the O'Donnell-Wimmer trick, which is a clever unorthodox way of generating a uniform random n-bit string, using random projections. The following lemma captures this trick:

Lemma 5.9. *First perform the projection step, where a $\boldsymbol{\rho} \sim \{y_{1/2}, 1_{1/2}\}^w \setminus \{1\}^w$ is sampled (i.e. we pick each y_i with probability $1/2$ and 1 with probability $1/2$, but never pick all 1's). Now perform a variable sampling step for the random variable $\boldsymbol{y} \sim \{0_{1-1/2^w}, 1_{1/2^w}\}$. Then doing the projection step and then the sampling step gives a uniform random n-bit string.*

Proof. The only way we could get the all 1's string is if we picked all 1's in the sampling step, and this happens with probability $1/2^w$. So the probability of not getting the all 1's string is $1 - 1/2^w$, and the only way of getting some other string (which is not the all 1's string) is if we have a version of the string that already has gone through y and then through $\boldsymbol{\rho}$, which happens with probability

$$\frac{1}{2^w - 1} \cdot \left(1 - \frac{1}{2^w}\right) = \frac{1}{2^w - 1} \cdot \frac{2^w - 1}{2^w} = \frac{1}{2^w}. \qquad \square$$

§5.3.2 Proof of the O'Donnell-Wimmer average-case lower bound

With all the setup in place, we can now prove the lemma that implies the result by O'Donnell and Wimmer:

Lemma 5.10. *Any CNF g that 0.2-approximates DNFTRIBES_n must have width*

$$\geqslant \frac{1}{4} \cdot 2^w = \Omega\left(\frac{n}{\log n}\right).$$

Proof. Consider a global version of the random sampling trick in Lemma 5.9 on an independent copy of $\boldsymbol{\rho}$ for each of the 2^w terms of DNFTRIBES_n. Recall that DNFTRIBES_n is the function

$$\text{DNFTRIBES}(x_1, \ldots, x_n) = (x_1 \wedge \ldots \wedge x_w) \vee (x_{w+1} \wedge \ldots \wedge x_{2w}) \vee \cdots \vee (x_{n-w+1} \wedge \ldots \wedge x_n),$$

where $n = w2^w$ and every variable occurs in exactly one term. In the draw of $\boldsymbol{\rho}_1$, $\boldsymbol{\rho}_2$, …, $\boldsymbol{\rho}_{2^w}$, for each $i \in [2^w]$, all the surviving variables under $\boldsymbol{\rho}_i$ become y_i. We now argue two things:

1. *The function* DNFTRIBES_n *"stays complex," that is, is balanced, under the random projection* $\boldsymbol{\rho}_i$ *for each* $i \in [2^w]$.

   Observe that with probability 1 over the random $\boldsymbol{\rho} = (\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_{2^w})$, the target function DNFTRIBES_n is balanced, because the restriction on the target function is then

   $$\text{DNFTRIBES}_n\restriction_{\boldsymbol{\rho}_1} = (y_1 \wedge \ldots \wedge y_w) \vee (y_{w+1} \wedge \ldots \wedge y_{2w}) \vee \cdots \vee (y_{n-w+1} \wedge \ldots \wedge y_n),$$

   which really just simplifies to

   $$\text{DNFTRIBES}_n\restriction_{\boldsymbol{\rho}_1} = y_1 \vee y_2 \vee \cdots \vee y_{2^w},$$

   by virtue of how the projection was set up. Now $\boldsymbol{y}_i \sim \{0_{1-1/2^w}, 1_{1/2^w}\}$, and so

   $$\mathbb{E}_{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{2^w}}\left[\text{DNFTRIBES}_n\restriction_{\boldsymbol{\rho}}(y)\right] = 1 - \left(1 - \frac{1}{2^w}\right)^{2^w} \geqslant 1 - \frac{1}{e} \approx 0.63.$$

2. *Any non-massively-wide CNF is very biased (either towards* 0 *or towards* 1*) after the application of the random projection* $\boldsymbol{\rho}$.

   Fix any CNF g' of width $\leqslant \frac{1}{4} \cdot 2^w$, and consider any fixed outcome $\boldsymbol{\rho}$ of the random projection. Consider $g'\restriction_{\boldsymbol{\rho}}$, which is a CNF over the variables $y_1, \ldots, y_{2^w}$. There are two possibilities here:

   a) Every clause of $g'\restriction_{\boldsymbol{\rho}}$ has $\geqslant 1$ negated variable:

      $$g'\restriction_{\boldsymbol{\rho}} = (\overline{y}_1 \vee \ldots) \wedge (\ldots \vee \overline{y_i} \vee \ldots) \wedge \ldots \wedge (\ldots \vee \overline{y_j} \vee \ldots).$$

      Then $g'\restriction_{\boldsymbol{\rho}}\left(0^{2^w}\right) = 1$ for the likely (with high probability $1 - 1/2^w$) input 0^{2^w}. But this isn't ideal, because the target function DNFTRIBES_n has $\text{DNFTRIBES}_n\restriction_{\boldsymbol{\rho}}(0^{2^w}) = 0$. Therefore

      $$\Pr\left[y = 0^{2^w}\right] = \left(1 - \frac{1}{2^w}\right)^{2^w} \approx \frac{1}{e} \approx 0.37,$$

      and so in this case $g'\restriction_{\boldsymbol{\rho}}$ and $\text{DNFTRIBES}_n\restriction_{\boldsymbol{\rho}}$ disagree on $37\% \geqslant 20\%$ of the y-outcomes.

   b) Not every clause of $g'\restriction_{\boldsymbol{\rho}}$ has $\geqslant 1$ negated variable:

      $$g'\restriction_{\boldsymbol{\rho}} = (y_1 \vee \ldots) \wedge (\ldots \vee \overline{y_i} \vee \ldots) \wedge \ldots \wedge (\ldots \vee y_j \vee \ldots),$$

      that is, there exists some clause $C = y_1 \vee y_2 \vee \ldots \vee y_k$ in $g'\restriction_{\boldsymbol{\rho}}$ with $k \leqslant \frac{1}{4} \cdot 2^w$ variables. But again we are in trouble, because

      $$\begin{aligned}\Pr_{\boldsymbol{y}}\left[g'\restriction_{\boldsymbol{\rho}}(y) = 1\right] &\leqslant \Pr_{\boldsymbol{y}}\left[C(y) = 1\right] \\ &\overset{\text{union bound}}{\leqslant} \sum_{i=1}^{2^w} \Pr_{\boldsymbol{y}}\left[y_i = 1\right] \leqslant \frac{1}{4} \cdot 2^w \cdot \frac{1}{2^w} = \frac{1}{4}.\end{aligned}$$

But

$$\Pr_{\boldsymbol{y}}\left[\text{DNFTRIBES}_n\restriction_{\boldsymbol{\rho}}(y)=1\right]=\Pr_{\boldsymbol{y}}\left[y_1\vee\ldots\vee y_{2^w}=1\right]=1-\left(1-\frac{1}{2^w}\right)^{2^w}\geqslant 1-\frac{1}{e}\approx 0.63,$$

and so in this case $g'\restriction_{\boldsymbol{\rho}}$ and $\text{DNFTRIBES}_n\restriction_{\boldsymbol{\rho}}$ disagree on $63\%-25\%\geqslant 20\%$ of the y-outcomes.

So overall, we have that the overall probability that they agree on a uniformly random $\boldsymbol{x}$ is, by the O'Donnell-Wimmer trick,

$$\begin{aligned}\Pr_{\boldsymbol{x}\sim\{0,1\}^n\text{ unif.}}\left[g(\boldsymbol{x})\neq\text{DNFTRIBES}_n(\boldsymbol{x})\right]&=\mathbb{E}_{\text{indep. copies of }\boldsymbol{\rho}}\left[\Pr_{\boldsymbol{y}}\left[g'\restriction_{\boldsymbol{\rho}}(\boldsymbol{y})\neq\text{DNFTRIBES}_n\restriction_{\boldsymbol{\rho}}(\boldsymbol{y})\right]\right]\\&\geqslant 0.2,\end{aligned}$$

for every $\boldsymbol{\rho}$, so g must have width $\geqslant\frac{1}{4}\cdot 2^w=\Omega\left(\frac{n}{\log n}\right)$, and we have proved Theorem 5.2 as a direct implication from this lemma. □

A small note: to defeat all depth-2 circuits, switch 0/1 and AND/OR in the above argument, and then we can use the same exact argument to get the following corollary:

Corollary 5.11. *Any DNF g' that* 0.1*-approximates the n-variable* CNFTRIBES_n *function, defined analogously as*

$$\text{CNFTRIBES}_n(x_1,\ldots,x_n)=(x_1\vee\ldots\vee x_w)\wedge(x_{w+1}\vee\ldots\vee x_{2w})\wedge\cdots\wedge(x_{n-w+1}\vee\ldots\vee x_n),$$

must have at least $2^{\Omega\left(\frac{n}{\log n}\right)}$ many terms.

So we know that CNFTRIBES_n is hard to approximate by a DNF, and DNFTRIBES_n is hard to approximate by a CNF. Can we cruelly cook up a hard function for both that requires computing DNFTRIBES_n and CNFTRIBES_n to compute it everywhere?

Claim 5.12. *Consider the function $T\colon\{0,1\}^{2w2^w}\to\{0,1\}$ defined as*

$$T(a,b)=\text{DNFTRIBES}_n(a)\vee\text{CNFTRIBES}_n(b).$$

Then T is hard to approximate by both DNFs and CNFs. In particular, any depth-2 circuit that 0.01 *approximates $T(a,b)$ must have size $2^{\Omega\left(\frac{n}{\log n}\right)}$.*

§5.4 Basics of $\mathbb{F}_2$-polynomials

We will now move on to a different kind of lower bound, which is for the class of $\mathbb{F}_2$-polynomials. We will first define what $\mathbb{F}_2$-polynomials are, and then we will see how to prove lower bounds for them.

Definition 5.13. *An $\mathbb{F}_2$-polynomial is a function $f\colon\{0,1\}^n\to\{0,1\}$ that can be written as*

$$f(x_1,\ldots,x_n)=\bigoplus_{S\subseteq[n]}a_S\cdot\bigotimes_{i\in S}x_i,$$

where $a_S\in\{0,1\}$, and $\bigoplus$ and $\bigotimes$ denote the $\mathbb{F}_2$-sum and $\mathbb{F}_2$-product, respectively.

A monomial over $\mathbb{F}_2$ is an expression of the form $\bigotimes_{i \in S} x_i$ where the i are distinct, and this is the same as the AND over the variables x_i for $i \in S$. The $\mathbb{F}_2$-sum is the XOR over the monomials. We will never need to consider powers of these monomials, because $x_i^n = x_i$ for all $x_i \in \{0,1\}$ and all $n \in \mathbb{N}$. So all monomials for us will be multilinear, with degree $\leqslant 1$ in each variable and total degree-k for the polynomial.

Notice that there are 2^n multilinear monomials, and therefore 2^{2^n} multilinear $\mathbb{F}_2$-polynomials; and remember that there are 2^{2^n} functions from $\{0,1\}^n \to \{0,1\}$, so every function from $\{0,1\}^n \to \{0,1\}$ can be written as an $\mathbb{F}_2$-polynomial.

Fact 5.14. *Every function $f\colon \{0,1\}^n \to \{0,1\}$ has a unique expression as an $\mathbb{F}_2$-polynomial.*

Proof. Easy exercise—too easy for homework. Just prove by induction. □

Our goal will be to prove degree lower bounds—that functions have to have high degree—which will be useful as we continue to discuss PRGs. Indeed,

$$\mathsf{AND}(x_1, \dots, x_n) = x_1 \cdot x_2 \cdot \dots \cdot x_n,$$

and so worst-case degree lower bounds are trivial, and we can't compute the AND function on a degree-$(n-1)$ $\mathbb{F}_2$-polynomial or lower. The hard problem is proving strong average-case (correlation) lower bounds for $\mathbb{F}_2$-polynomials.

Let

$$\mathrm{DEG}_d \coloneqq \{\text{all functions } f : \{0,1\}^n \to \{0,1\} \text{ that have degree-}d\ \mathbb{F}_2\text{-polynomials}\}.$$

We know that $\mathsf{AND} \notin \mathrm{DEG}_{n-1}$, and like we just mentioned, average-case lower bounds are hard to prove. In fact, here's something nobody knows how to do:

Open Problem 1. *Prove that some explicit function $f\colon \{0,1\}^n \to \{0,1\}$ corresponding to some language in NP is $1/n$-hard for $\log n$-degree polynomial threshold functions for some distribution $\mathcal{D}$ over $\{0,1\}^n$.*

Suppose this can't be done. Then for every distribution $\mathcal{D}$, there is a low-degree $\mathbb{F}_2$-polynomial that has this advantage and can serve as a weak learner for which we can run a boosting algorithm; this would imply that there is a poly-sized circuit with a very simple structure—$\mathsf{MAJ}\colon \{0,1\}^n \to \{0,1\}$ of degree $\log n$—that would compute any function in NP, which would then imply that $\mathsf{NP} \subseteq \mathsf{P/poly}$, and consequently $\mathsf{P} = \mathsf{NP}$. So if we believe that NP doesn't have (quasi-)poly-sized circuits, then we should believe that this open problem is doable.

Although they are hard in general, next time we will do two correlation bounds:

1. High degree (in fact, degree $\Theta(\sqrt{n})$), and constant correlation $\Theta(1)$;
2. Low degree (in fact, $\ll \log n$), but tiny correlation.

§6 Lecture 06—20th February, 2024

Last time. In the previous class, we finished the unit on constant-depth circuits and started a new unit on $\mathbb{F}_2$-polynomials. In particular, we:

- Finished the proof of the Håstad switching lemma by proving a key fact about the Angel of a restriction $\boldsymbol{\sigma}$.
- Gave average-case lower bounds for AC^0 circuits for PARITY_n, showing that depth-d size-$2^{cn^{1/d}}$ circuits for PARITY_n cannot achieve accuracy $\geqslant 1/2 + 1/2^{cn^{1/d}}$.
- Gave O'Donnell-Wimmer depth-2 average case lower bounds, showing that depth-2 size-$2^{cn/\log n}$ circuits cannot achieve accuracy $\geqslant 90\%$ for the function $\mathrm{DNFTRIBES}(a) \wedge \mathrm{CNFTRIBES}(b)$ using random projections.
- Started a unit on $\mathbb{F}_2$-polynomials, introducing the notion of $\mathbb{F}_2$-polynomials and the $\mathbb{F}_2$-degree of a function; also saw the following unsolved challenge:

  > Prove that some explicit function $f\colon \{0,1\}^n \to \{0,1\}$ corresponding to some language in NP is $1/n$-hard for $\log n$-degree polynomial threshold functions for some distribution $\mathcal{D}$ over $\{0,1\}^n$.

Today. We will see two results towards the above challenge:

- Any degree-$\left(\frac{1}{4}\sqrt{n}\right)$ $\mathbb{F}_2$-polynomial p has

$$\Pr_{\boldsymbol{x}\sim\mathcal{U}}\left[\mathrm{MOD}_3(\boldsymbol{x}) = p(\boldsymbol{x})\right] \leqslant \frac{7}{8}.$$

- Any degree-d $\mathbb{F}_2$-polynomial p has

$$\Pr_{\boldsymbol{x}\sim\mathcal{U}}\left[\mathrm{GIP}_{d+1}(\boldsymbol{x}) = p(\boldsymbol{x})\right] \leqslant \frac{1}{2} + 2^{-\Omega\left(\frac{n}{d\cdot 2^d}\right)},$$

  where $\mathrm{GIP}\colon \{0,1\}^n \to \{0,1\}$ is the generalised inner product.
- Start basic tools for derandomisation and PRGs.

§6.1 Smolensky's weak correlation bound for fairly-high-degree $\mathbb{F}_2$-polynomials

For our first foray into correlation bounds for $\mathbb{F}_2$-polynomials, we will see one for which there is high degree but a weak bound on correlation. First we define the following function

Definition 6.1. *The function* $\mathrm{MOD}_3\colon \{0,1\}^n \to \{0,1\}$ *is defined as*

$$\mathrm{MOD}_3(x) = \begin{cases} 1 & x_1 + x_2 + \ldots + x_n \equiv 1 \pmod 3, \\ 0 & \textit{otherwise.} \end{cases}$$

This is a natural Boolean function, with applications in error detection and correction.

Example 6.2. We have that $\text{MOD}_3(1000) = 1$ and $\text{MOD}_3(1111) = 1$, while $\text{MOD}_3(1100) = 0$ and $\text{MOD}_3(1110) = 0$.

We will prove the following version of the result by Smolensky:

Theorem 6.1 (Smolensky, [Smo87])**.** *Any degree-$(\varepsilon\sqrt{n})$ $\mathbb{F}_2$-polynomial p has, with $\varepsilon = 1/4$,*

$$\Pr_{\boldsymbol{x}\sim\mathcal{U}}[\text{MOD}_3(\boldsymbol{x}) = p(\boldsymbol{x})] \leqslant \frac{7}{8},$$

that is, $\text{Cor}\left[\text{MOD}_3, \text{DEG}_{\varepsilon\sqrt{n}}\right] \leqslant \frac{3}{4}$.

The high-level idea of the proof is as follows. Fix any candidate $\mathbb{F}_2$-polynomial p of degree $d = \varepsilon\sqrt{n}$. Define X to be the set of all bit vectors where p is correct:

$$X \coloneqq \{x \in \{0,1\}^n : p(x) = \text{MOD}_3(x)\}.$$

Our goal in this proof is to argue that $|X|$ is not too big; in particular, that $|X| \leqslant \frac{7}{8} \cdot 2^n$.

Intuition Intuitively, it seems fairly plausible that MOD_3 should have high-degree as an $\mathbb{F}_2$-polynomial. This should mean that perhaps MOD_3 can be used to simulate a high-degree monomial and consequently all high-degree $\mathbb{F}_2$-polynomials over $\{0,1\}^n$. So if p agrees with MOD_3 on a large fraction of inputs, then given p, we may use it to simulate high-degree $\mathbb{F}_2$-polynomials over X. But since p has low degree, it shouldn't be possible to use a low-degree polynomial to simulate a high-degree polynomial over all of X; so if p can simulate a high-degree function everywhere on X and $|X|$ is really large, then p must have high degree, which shouldn't be possible. So $|X|$ shouldn't be too big.

The proof will use an extension field $\mathbb{F}$ of $\mathbb{F}_2$, with $|\mathbb{F}| = 4$. We will use the degree-d polynomial p to express any function from X to $\mathbb{F}$, using an $\mathbb{F}$-polynomial of degree $d + n/2$. Then we're essentially done, because this means that the number of functions mapping X to $\mathbb{F}$ must be at most the number of $\mathbb{F}$-polynomials of degree $d + n/2$, that is, $4^{|X|}$ is at most the number of $\mathbb{F}$-polynomials of degree $d + n/2$.

Setup Assume, without loss of generality, that n is divisible by 3; if it isn't, ignore the last variable and replace n by $n - 1$. Then

$$\text{MOD}_3(1 + x_1, 1 + x_2, \ldots, 1 + x_n) = 1,$$

if and only if $3k + 1$ for some $k \in \mathbb{N}$ of the $1 + x_i$'s are 1, which happens if and only if $3k + 1$ of the x_i's are 0, and if n is divisible by 3, then $3k + 1$ of the x_i's are 0 if and only if $3\ell + 2$ for some $\ell \in \mathbb{N}$ of the x_i's are 1. So we can write

$$\sum_{i=1}^{n} x_i \equiv 2 \pmod 3.$$

Let $\mathbb{F}$ be the 4-element field obtained as $\mathbb{F}_2[t]/(t^2 + t + 1)$, the ring of polynomials $\mathbb{F}_2[t]$ over $\mathbb{F}_2$ in the variable t taken modulo the ideal $(t^2 + t + 1)$, generated by the irreducible polynomial $t^2 + t + 1$. Thus $\mathbb{F}$ is the four element field $\{0, 1, t, 1 + t\}$, with addition and multiplication defined as in $\mathbb{F}_2[t]$ and reducing modulo $t^2 + t + 1$. So we are working modulo 2 and $t^2 + t + 1$, with $t^2 + t + 1 = 0 \implies t^2 = t + 1$.

Example 6.3. We have that $t^3 = t \cdot (t^2) = t \cdot (t+1) = t^2 + t = 1$.

Now define $h\colon \{1, t\} \to \mathbb{F}_2$ as the *change-of-domain* (affine) map (i.e a degree-1 polynomial) given by

$$h(\alpha) = \frac{\alpha + 1}{t + 1} = t \cdot (\alpha + 1) = t\alpha + t.$$

Then $h(1) = 0$ and $h(t) = 1$. We can now prove our first claim that lives up to the intuition we presented earlier:

Claim 6.4. *For any $y \in \{1, t\}^n$, we have*

$$\underbrace{y_1 y_2 \cdots y_n}_{\text{polynomial in } t} = 1 + (t+1) \cdot \underbrace{\textsc{mod}_3\left(h(y_1), h(y_2), \ldots, h(y_n)\right)}_{0 \text{ or } 1} + (t^2+1) \cdot \underbrace{\textsc{mod}_3\left(1 + h(y_1), 1 + h(y_2), \ldots, 1 + h(y_n)\right)}_{0 \text{ or } 1}.$$

Proof. Recall the definition of the function $\textsc{mod}_3$:

$$\textsc{mod}_3(h(y_1), h(y_2), \ldots, h(y_n)) = \begin{cases} 1 & \text{if } \sum_{i=1}^n h(y_i) \equiv 1 \pmod 3, \\ 0 & \text{otherwise.} \end{cases}$$

By the first paragraph of our setup,

$$\textsc{mod}_3(1 + h(y_1), 1 + h(y_2), \ldots, 1 + h(y_n)) = \begin{cases} 1 & \text{if the number of } t\text{'s in } y \text{ is} \equiv 2 \pmod 3, \\ 0 & \text{otherwise.} \end{cases}$$

Working casewise, we have that

1. When the number of t's in y is $\equiv 0 \pmod 3$, the left hand side is $t^{3k} = 1$ since $t^3 = 1$, and the right hand side is $1 + (t+1) \cdot 0 + (t^2+1) \cdot 0 = 1$.
2. When the number of t's in y is $\equiv 1 \pmod 3$, the left hand side is $t^{3k+1} = t$ since $t^3 = 1$, and the right hand side is $1 + (t+1) \cdot 1 + (t^2+1) \cdot 0 = t$.
3. When the number of t's in y is $\equiv 2 \pmod 3$, the left hand side is $t^{3k+2} = t^2$ since $t^3 = 1$, and the right hand side is $1 + (t+1) \cdot 0 + (t^2+1) \cdot 1 = t^2$. □

So we have figured out that we can express any high-degree monomial in terms of the $\textsc{mod}_3$ function using the change-of-domain map h, and before that, we argued that the $\textsc{mod}_3$ function should have high degree. So we can now start talking directly about p itself, and prove the main result here.

Proof of Theorem 6.1. Fix p to be any $\mathbb{F}_2$-polynomial from $\{0,1\}^n$ to $\{0,1\}$ of degree $d = \varepsilon\sqrt{n}$. Let

$$\delta = \Pr_{\boldsymbol{u} \sim \mathbb{F}_2^n}\left[p(\boldsymbol{u}) \neq \textsc{mod}_3(\boldsymbol{u})\right].$$

The goal of our proof will be to show that $\delta \geqslant \frac{1}{8}$, so that $\Pr_{\boldsymbol{u} \sim \mathcal{U}}\left[p(\boldsymbol{u}) = \textsc{mod}_3(\boldsymbol{u})\right] \leqslant \frac{7}{8}$.

Let $p' : \{1, t\} \to \mathbb{F}$ be the polynomial defined by

$$p'(y) = 1 + (t+1) \cdot p\left(h(y_1), h(y_2), \ldots, h(y_n)\right) + (t^2+1) \cdot p\left(1 + h(y_1), 1 + h(y_2), \ldots, 1 + h(y_n)\right).$$

Observe that if p were the exact same as $\textsc{mod}_3$, then p' would be exactly $y_1 y_2 \cdots y_n$ in $\mathbb{F}$ via Claim 6.4. But presumably it's not; they disagree on some inputs, so if we focus on how often they disagree we may be able to say something about δ.

Via Claim 6.4, we have that

$$\Pr_{\boldsymbol{y} \sim \{1,t\}^n} \left[p'(y_1, y_2, \ldots, y_n) \neq y_1 y_2 \cdots y_n\right] \geqslant 1 - 2\delta,$$

because both $h(y_1, y_2, \ldots, y_n)$ and $(1 + h(y_1), 1 + h(y_2), \ldots, 1 + h(y_n))$ are uniformly distributed over $\{0,1\}^n$ when $\boldsymbol{y}$ is uniformly distributed over $\{1, t\}^n$.

Now p, by assumption, is a degree-d polynomial and h is a degree-1 polynomial, so p' is also a degree-d polynomial. In particular, since $\deg(h) = 1$ and $\deg(p) = d$, we have that $\deg(p') \leqslant d$. Define $S \subseteq \{1, t\}^n$ to be

$$S = \left\{y \in \{1, t\}^n : y_1 y_2 \cdots y_n = p'(y_1, y_2, \ldots, y_n)\right\}.$$

We just saw that $|S| \geqslant 2^n (1 - 2\delta)$, from the fact that p' and $y_1 y_2 \cdots y_n$ disagree on at most 2δ fraction of inputs.

Consider any function $f \colon S \to \mathbb{F}$. We can write any such f as a multilinear polynomial:

$$f(y) = \sum_{(a_1, \ldots, a_n) \in \{1,t\}^n} f(a_1, \ldots, a_n) \cdot \prod_{i=1}^{n} \left(1 + h(y_i) + h(a_i)\right).$$

The key claim for us is that any multilinear monomial M over $y_1, \ldots, y_n$ of degree $\geqslant n/2$ can be reexpressed as a polynomial of degree $\leqslant d + n/2$, without affecting the value on any string in S. Our monomial with degree $\geqslant n/2$ is then

$$\begin{aligned}
\prod_{i \in M} y_i &= y_1 \cdots y_n \cdot \prod_{i \notin M} (y_i t + y_i + t) \\
&= p'(y_1, y_2, \ldots, y_n) \cdot \prod_{i \notin M} (y_i t + y_i + t) \qquad \text{since we are considering } y \in S.
\end{aligned}$$

We now show that this particular representation holds. Note that if $i \in M$, then it's on both sides of the equality, and if $i \notin M$, then there is no y_i on the left hand side (i.e $y_i = 1$), and on the right hand side, we have $y_i \cdot (y_i t + y_i + t)$. There are then two cases; if $y_i = 1$, then the term is $1 \cdot (1 + t + t) = 1$, and if $y_i = t$, then the term is $t \cdot (t^2 + t + t) = t \cdot (1 + t) = t^2 + t = 1$. So the representation holds, and hence we can write any multilinear monomial of degree $\geqslant n/2$ as a polynomial of degree $\leqslant d + n/2$.

Do this for every monomial in the polynomial representation for f and write $f \colon S \to \mathbb{F}$ as a polynomial over $\mathbb{F}$ of degree $\leqslant d + n/2$. Now we know that there are $4^{|S|}$ functions from S to $\mathbb{F}$,

and there are $|F|^{\sum_{i=0}^{n/2+d}\binom{n}{i}} = 4^{\sum_{i=0}^{n/2+d}\binom{n}{i}}$ such polynomials over $\mathbb{F}$ of degree $\leqslant d + n/2$. So

$$\begin{aligned} 2^n(1-2\delta) \leqslant |S| \leqslant \sum_{i=0}^{n/2+d}\binom{n}{i} &\leqslant 2^{n-1} + d\cdot\binom{n}{n/2} \\ &\leqslant 2^n\cdot\left(\frac{1}{2}+\frac{d}{\sqrt{n}}\right) \\ &= 2^n\cdot\left(\frac{1}{2}+\varepsilon\right). \end{aligned}$$

So $1-2\delta \leqslant \frac{1}{2}+\varepsilon$, and hence $\delta \geqslant \frac{1}{4}-\frac{\varepsilon}{2}$, so that with $\varepsilon = 1/4$, we have $\delta \geqslant \frac{1}{8}$, and hence $\Pr_{\boldsymbol{u}\sim\mathcal{U}}[p(\boldsymbol{u}) = \text{MOD}_3(\boldsymbol{u})] \leqslant \frac{7}{8}$. □

§6.2 BNS's strong correlation bound for fairly-low-degree $\mathbb{F}_2$-polynomials

Now we focus on showing the second result:

Theorem 6.2 (Babai, Nisan, Szegedy, [BNS91])**.** *Any degree-d $\mathbb{F}_2$-polynomial p has*

$$\Pr_{\boldsymbol{x}\sim\mathcal{U}}[\text{GIP}_{d+1}(\boldsymbol{x}) = p(\boldsymbol{x})] \leqslant \frac{1}{2} + 2^{-\Omega\left(\frac{n}{d\cdot 2^d}\right)},$$

where $\text{GIP}\colon \{0,1\}^n \to \{0,1\}$ *is the generalised inner product.*

Intuition We can show, and in fact this is an official homework problem, that the inner product function $\text{IP}\colon \{0,1\}^n \to \{0,1\}$ defined as

$$\text{IP}(x) = \sum_{i=1}^{n-1} x_i x_{i+1} \pmod 2$$

has very low correlation with any degree-1 $\mathbb{F}_2$-polynomial, i.e. the PARITY_n function on any number of variables. Indeed, we can show that if $\deg(p) = 1$, then

$$\Pr_{\boldsymbol{x}\sim\mathcal{U}}[\text{IP}(\boldsymbol{x}) = p(\boldsymbol{x})] = \frac{1}{2} \pm \frac{1}{2^{n/2}}.$$

This is a very strong lower bound against degree-1 $\mathbb{F}_2$-polynomials; we'd like to do something more interesting, so what's hard for degree-2? Since the hardness comes from the quadraticity of the terms x_1x_2, x_2x_3, …, $x_{n-1}x_n$, we can guess that maybe the hard function for degree-2 $\mathbb{F}_2$-polynomials is something like $x_1x_2x_3 + x_4x_5x_6 + \cdots + x_{n-2}x_{n-1}x_n$. This motivates the definition of the generalised inner product function:

Definition 6.5. *The generalised inner product function* $\text{GIP}\colon \{0,1\}^n \to \{0,1\}$ *is defined as*

$$\text{GIP}_{d+1}(x) = \sum_{i=1}^{n-d} x_i x_{i+1}\cdots x_{i+d} \pmod 2.$$

This will be our hard function.

Setup Given a function $f\colon \mathbb{F}_2^n \to \{0,1\}$, write $\mathsf{e}(f) \coloneqq (-1)^f$, so that $\mathsf{e}(0) = 1$ and $\mathsf{e}(1) = -1$. Let $F(x) \coloneqq \mathsf{e}(f(x))$ for any $f\colon \mathbb{F}_2^n \to \{0,1\}$, so that $F\colon \mathbb{F}_2^n \to \{-1,1\}$. Recall that if $F, G : \mathbb{F}_2^n \to \{-1,1\}$, then

$$\operatorname{Cor}[F,G] = \left| \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{U}} [F(\boldsymbol{x})G(\boldsymbol{x})] \right|.$$

Our goal will be to analyse $\operatorname{Cor}[F, \mathrm{DEG}_d]$ for any d and $F\colon \mathbb{F}_2^n \to \{-1,1\}$. The helpful fact here is that the square of the correlation is expressible as the sum of functions related to F, and degree-$(d-1)$ $\mathbb{F}_2$-polynomials, effectively accomplishing some degree reduction (up to the additive function). Let us explore this: fix any $F\colon \mathbb{F}_2^n \to \{-1,1\}$, where $F = \mathsf{e}(f)$ for some $f\colon \mathbb{F}_2^n \to \{0,1\}$. Also, fix any degree-$d$ polynomial $p\colon \mathbb{F}_2^n \to \{0,1\}$, where $P = \mathsf{e}(p)$. Then the correlation between F and P is

$$\operatorname{Cor}[F,P] = \left| \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{U}} [F(\boldsymbol{x})P(\boldsymbol{x})] \right|.$$

Squaring this correlation yields:

$$\begin{aligned}
\operatorname{Cor}[F,P]^2 &= \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{U}} [F(\boldsymbol{x})P(\boldsymbol{x})]^2 \\
&= \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{U}} [F(\boldsymbol{x})P(\boldsymbol{x})] \cdot \mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{U}} [F(\boldsymbol{y})P(\boldsymbol{y})] \\
&= \mathop{\mathbb{E}}_{\boldsymbol{x},y\sim\mathcal{U}} [F(\boldsymbol{x})P(\boldsymbol{x})F(\boldsymbol{y})P(\boldsymbol{y})],
\end{aligned}$$

by the independence of $\boldsymbol{x}$ and $\boldsymbol{y}$. Now since $\boldsymbol{x}$ is a uniform random variable and $\boldsymbol{y}$ is an independent uniform random variable, we can write $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{h}$ for some independent $\boldsymbol{h} \sim \mathcal{U}$, and hence

$$Cor[F,P]^2 = \mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{h}\sim\mathcal{U}} [F(\boldsymbol{x})P(\boldsymbol{x})F(\boldsymbol{x}+\boldsymbol{h})P(\boldsymbol{x}+\boldsymbol{h})].$$

Note now that by the definition of P,

$$\begin{aligned}
P(x)P(x+h) &= \mathsf{e}(p(x)) \cdot \mathsf{e}(p(x+h)) \\
&= \mathsf{e}(p(x) + p(x+h))
\end{aligned}$$

Our key observation here is that for any fixed $\boldsymbol{h}$, for p a degree-d polynomial, the function $p(x) + p(x+h)$ is a degree-$(d-1)$ polynomial.

Example 6.6. If $p = x_1 \cdots x_d$, then

$$p(x) + p(x+h) = x_1 \cdots x_d + (x_1 + h_1) \cdots (x_d + h_d) = x_1 \cdots x_d + x_1 \cdots x_d + l,$$

where l constitutes a series of lower degree polynomial terms.

Thus the squared correlation $\operatorname{Cor}[F,P]^2$ is equal to the average of the correlations of $F(\boldsymbol{x})F(\boldsymbol{x}+\boldsymbol{h})$ with a degree-$(d-1)$ polynomial, and we have made progress!

Now write $F^{+y}(x)$ for $F(x+y)$. The following definition from additive combinatorics will be very friendly towards us:

Definition 6.7 (Gowers uniformity)**.** *Let $F : \mathbb{F}_2^n \to [-1,1]$, and pick an integer $k \geqslant 0$. Then the k-uniformity of f is*

$$\mathsf{U}_k(F) = \mathop{\mathbb{E}}_{\boldsymbol{h}_1,\ldots,\boldsymbol{h}_k,\boldsymbol{x}\sim\mathcal{U}} \left[\prod_{S\subseteq[k]} F^{+\sum_{j\in S} \boldsymbol{h}_j}(\boldsymbol{x}) \right].$$

The interpretation of this is as follows. Any fixed $h_1, \dots, h_k$ defines a k-dimensional hypercube in $\mathbb{F}_2^n$, and the value of $F^{+\sum_{j\in S} h_j}(\boldsymbol{x})$ is the value of F on the k-dimensional hypercube translated by $\sum_{j\in S} h_j$. The k-uniformity of F is then the average of the product of the values of F on all such Boolean hypercubes, over all possible choices of $h_1, \dots, h_k$ and x.

Example 6.8. If $k = 0$, then there are no h_i's, and the 0-uniformity is $\mathsf{U}_0(F) = \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{U}}[F(\boldsymbol{x})]$. If $k = 1$, then there is one h_1, and the 1-uniformity is

$$\begin{aligned}\mathsf{U}_1(F) &= \mathbb{E}_{\boldsymbol{h}_1\sim\mathcal{U}}\left[\mathsf{U}_0(F\cdot F^{+\boldsymbol{h}_1})\right]\\ &= \mathop{\mathbb{E}}_{\boldsymbol{h}_1,\boldsymbol{x}\sim\mathcal{U}}\left[F(\boldsymbol{x})F^{+h_1}(\boldsymbol{x})\right]\\ &= \mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{U}}[F(\boldsymbol{x})F(\boldsymbol{y})] = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}}[F(\boldsymbol{x})]^2 \geqslant 0.\end{aligned}$$

If $k = 2$, then there are two $\boldsymbol{h}_i$'s, and the 2-uniformity is

$$\begin{aligned}\mathsf{U}_2(F) &= \mathbb{E}_{\boldsymbol{h}_2\sim\mathcal{U}}\left[\mathsf{U}_1(F\cdot F^{+\boldsymbol{h}_2})\right]\\ &= \mathbb{E}_{\boldsymbol{h}_1,\boldsymbol{h}_2\sim\mathcal{U}}\left[\mathsf{U}_0(F\cdot F^{+\boldsymbol{h}_1}\cdot F^{+\boldsymbol{h}_2}\cdot F^{+\boldsymbol{h}_1+\boldsymbol{h}_2})\right]\\ &= \mathop{\mathbb{E}}_{\boldsymbol{h}_1,\boldsymbol{h}_2,\boldsymbol{x}\sim\mathcal{U}}\left[F(\boldsymbol{x})F^{+\boldsymbol{h}_1}(\boldsymbol{x})F^{+\boldsymbol{h}_2}(\boldsymbol{x})F^{+\boldsymbol{h}_1+\boldsymbol{h}_2}(\boldsymbol{x})\right] \geqslant 0.\end{aligned}$$

In general,

$$\mathsf{U}_{k+1}(F) = \mathbb{E}_{\boldsymbol{h}_{k+1}\sim\mathcal{U}}\left[\mathsf{U}_k(F\cdot F^{+\boldsymbol{h}_{k+1}})\right].$$

Lemma 6.9 ($(d+1)$-uniformity of AND)**.** *Let $F = \mathsf{e}(f)$, let the AND function be $f\colon \mathbb{F}_2^{d+1} \to \{0,1\}$ defined as $f = x_1x_2\cdots x_{d+1}$, so that F outputs -1 on the input 1^{d+1} and 1 on all other inputs. Then the $(d+1)$-uniformity of F is $\mathsf{U}_{d+1}(F) \approx 0.6$.*

Proof. We claim that $\mathsf{U}_{d+1}(F) = 1 - 2p$, where

$$p = \Pr_{\boldsymbol{x},\boldsymbol{h}_1,\dots,h_j\sim\mathcal{U}}\left[\prod_{S\subseteq[d+1]} F^{+\sum_{j\in S}\boldsymbol{h}_j}(\boldsymbol{x}) = -1\right].$$

Now if we have

$$\begin{aligned}h_1 = e_1 &= (1,0,0,\dots,0),\\ h_2 = e_2 &= (0,1,0,\dots,0),\\ &\vdots\\ h_{d+1} = e_{d+1} &= (0,0,0,\dots,1),\end{aligned}$$

for the standard basis vectors e_i, then we have that regardless of the specific values of x_i, the sum $\sum_{j\in S} h_i$ varies over all possible vectors in $\mathbb{F}_2^{d+1}$, and hence the product $\prod_{S\subseteq[d+1]} F^{+\sum_{j\in S}\boldsymbol{h}_j}(\boldsymbol{x})$ is

the product of $F(z)$ as some z varies over all of $\mathbb{F}_2^{d+1}$ (in fact, this is true for any basis $h_1, \ldots, h_{d+1}$). The function F is -1 once, so

$$\prod_{\text{all } z \in \mathbb{F}_2^{d+1}} F(z) = -1,$$

On the other hand, if $h_1, \ldots, h_{d+1}$ is not a basis, then for any x, the arguments to $F^{+\sum_{j\in S} \boldsymbol{h}_j}(x)$ each occur an even number of times, and $\prod_{S\subseteq[d+1]} F^{+\sum_{j\in S} \boldsymbol{h}_j}(x) = 1$. Consequently,

$$\begin{aligned} p &= \Pr_{\boldsymbol{h}_1,\ldots,\boldsymbol{h}_{d+1}} \left[\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{d+1} \text{ basis of } F_2^{d+1}\right] \\ &= \left(1 - \frac{1}{2^{d+1}}\right) \cdot \left(1 - \frac{2}{2^{d+1}}\right) \cdot \ldots \cdot \left(1 - \frac{2^d}{2^{d+1}}\right) \\ &= \frac{1}{2} \cdot \frac{3}{4} \cdot \ldots \cdot \frac{2^{d+1}-1}{2^{d+1}} \\ &\approx 0.2. \end{aligned}$$

So $\mathsf{U}_{d+1}(F) = 1 - 2p \approx 0.6$. □

Now here's a lemma:

Lemma 6.10. *Let $F\colon \mathbb{F}_2^n \to \{-1,1\}$ be any function, and let $p\colon \mathbb{F}_2^n \to \{0,1\}$ be a degree-d polynomial, and let $P = \mathsf{e}(p)$. Then*

$$\mathrm{Cor}[F,P] \leqslant \mathsf{U}_{d+1}(F)^{\frac{1}{2^{d+1}}}.$$

A fact that is immediate from the definition of Gowers uniformity is as follows:

Fact 6.11. *Let $F_1, F_2 : \mathbb{F}_2^n \to \{-1,1\}$. Define $G(x,y) = F_1(x)F_2(y)$ to be their product over disjoint sets of inputs. Then $\mathsf{U}_k(G) = \mathsf{U}_k(F_1)\mathsf{U}_k(F_2)$.*

Proof. We have that

$$\begin{aligned} \mathsf{U}_k(G) &= \mathbb{E}_{\boldsymbol{h}_1,\ldots,\boldsymbol{h}_k,\boldsymbol{x},\boldsymbol{y}\sim\mathcal{U}} \left[\prod_{S\subseteq[k]} G^{+\sum_{j\in S} \boldsymbol{h}_j}(\boldsymbol{x},\boldsymbol{y})\right] \\ &= \mathbb{E}_{\boldsymbol{h}_1,\ldots,\boldsymbol{h}_k,\boldsymbol{x},\boldsymbol{y}\sim\mathcal{U}} \left[\prod_{S\subseteq[k]} F_1^{+\sum_{j\in S} \boldsymbol{h}_j}(\boldsymbol{x}) F_2^{+\sum_{j\in S} \boldsymbol{h}_j}(\boldsymbol{y})\right] \\ &= \mathbb{E}_{\boldsymbol{h}_1,\ldots,\boldsymbol{h}_k,\boldsymbol{x}\sim\mathcal{U}} \left[\prod_{S\subseteq[k]} F_1^{+\sum_{j\in S} \boldsymbol{h}_j}(\boldsymbol{x})\right] \cdot \mathbb{E}_{\boldsymbol{h}_1,\ldots,\boldsymbol{h}_k,\boldsymbol{y}\sim\mathcal{U}} \left[\prod_{S\subseteq[k]} F_2^{+\sum_{j\in S} \boldsymbol{h}_j}(\boldsymbol{y})\right] \\ &= \mathsf{U}_k(F_1)\mathsf{U}_k(F_2). \end{aligned}$$

□

Combining all of these results, we now have the main theorem by Babai, Nisan, and Szegedy:

Theorem 6.3. *Let $\mathrm{GIP}_{m,d+1}$ be the degree-$(d+1)$ polynomial over $\mathbb{F}_2^n$, where $n = m(d+1)$:*

$$\mathrm{GIP}_{m,d+1}(x) = \underbrace{x_1 \cdots x_{d+1} + \ldots + x_{n-d} \cdots x_n}_{n \text{ monomials}}.$$

Then,

$$\mathrm{Cor}\left[\mathsf{e}\left(\mathrm{GIP}_{m,d+1}\right), \mathsf{e}\left(\mathrm{DEG}_d\right)\right] \leqslant \exp\left(-\Omega\left(\frac{n}{d \cdot 2^d}\right)\right).$$

Proof. The correlation is upper bounded by

$$\begin{aligned}
\mathrm{Cor}\left[\mathsf{e}\left(\mathrm{GIP}_{m,d+1}\right), \mathsf{e}\left(\mathrm{DEG}_d\right)\right] &\leqslant \mathsf{U}_{d+1}(e\left(\mathrm{GIP}_{m,d+1}\right))^{\frac{1}{2^{d+1}}} && \text{by Lemma 6.10} \\
&= \mathsf{U}_{d+1}\left(\mathsf{e}\left(\mathsf{AND}_{d+1}\right)\right)^{\frac{m}{2^{d+1}}} && \text{by Fact 6.11} \\
&\leqslant (0.6)^{\frac{m}{2^{d+1}}},
\end{aligned}$$

by Lemma 6.9. This is the desired result. □

Next time we will discuss tools for PRGs.

§7 Lecture 07—27th February, 2024

Last time. In the previous class, we saw two incomparable correlation bounds for $\mathbb{F}_2$-polynomials:

- Any degree-$\left(\frac{1}{4}\sqrt{n}\right)$ $\mathbb{F}_2$-polynomial p has

$$\Pr_{\boldsymbol{x}\sim\mathcal{U}}\left[\mathrm{MOD}_3(\boldsymbol{x}) = p(\boldsymbol{x})\right] \leqslant \frac{7}{8}.$$

- Any degree-d $\mathbb{F}_2$-polynomial p has

$$\Pr_{\boldsymbol{x}\sim\mathcal{U}}\left[\mathrm{GIP}_{d+1}(\boldsymbol{x}) = p(\boldsymbol{x})\right] \leqslant \frac{1}{2} + 2^{-\Omega\left(\frac{n}{d\cdot 2^d}\right)},$$

  where $\mathrm{GIP}\colon \{0,1\}^n \to \{0,1\}$ is the generalised inner product.

Today. We will:

- Finish the proof of the missing piece from the second correlation bound:

  Let $F\colon \mathbb{F}_2^n \to \{-1,1\}$ be any function, and let $p\colon \mathbb{F}_2^n \to \{0,1\}$ be a degree-d polynomial, and let $P = \mathsf{e}(p)$. Then

$$\mathrm{Cor}[F,P] \leqslant \mathsf{U}_{d+1}(F)^{\frac{1}{2^{d+1}}}.$$

- Start the basic tools for PRGs:
  - k-wise independence of random variables and why they exist.
  - ε-biased random variables.
  - k-wise independent ε-biased random variables.
  - some Fourier analysis over the Boolean hypercube, and some applications.

§7.1 Finishing the proof of Babai-Nisan-Szegedy's strong correlation bound

Towards a proof of Lemma 6.10, we present the following facts:

Fact 7.1. *For all functions $F\colon \mathbb{F}_2^n \to \{-1,1\}$, we have $\sqrt{\mathsf{U}_1(F)} = |\mathbb{E}[F]|$.*

Proof. See Example 6.8. □

Fact 7.2. *For any function $F\colon \mathbb{F}_2^n \to \{-1,1\}$, we have $\mathsf{U}_k(F) \leqslant \sqrt{\mathsf{U}_{k+1}(F)}$.*

Proof. We will prove this in the case $k = 1$; the others follow by induction. We have

$$\begin{aligned}
\mathsf{U}_2(F) &= \mathbb{E}_{\boldsymbol{h}\sim\mathcal{U}}\left[\mathbb{E}_{\boldsymbol{x},\boldsymbol{h}_2\sim\mathcal{U}}\left[F(\boldsymbol{x})F^{+\boldsymbol{h}_1}(\boldsymbol{x})F^{\boldsymbol{h}_2}(\boldsymbol{x})F^{+\boldsymbol{h}_1+\boldsymbol{h}_2}(\boldsymbol{x})\right]\right] \\
&= \mathbb{E}_{\boldsymbol{h}_1\sim\mathcal{U}}\left[\mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}}\left[F(\boldsymbol{x})F^{+\boldsymbol{h}_1}(\boldsymbol{x})\right]\cdot\mathbb{E}_{\boldsymbol{y}\sim\mathcal{U}}\left[F(\boldsymbol{y})F^{+\boldsymbol{h}_1}(\boldsymbol{y})\right]\right] \\
&= \mathbb{E}_{\boldsymbol{h}_1\sim\mathcal{U}}\left[\mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}}\left[F(\boldsymbol{x})F^{+\boldsymbol{h}_1}(\boldsymbol{x})\right]^2\right] \\
&\geqslant \left(\mathbb{E}_{\boldsymbol{h}_1,\boldsymbol{x}\sim\mathcal{U}}\left[F(\boldsymbol{x})F^{+\boldsymbol{h}_1}(\boldsymbol{x})\right]\right)^2 && \text{by Cauchy-Schwarz} \\
&= \mathsf{U}_1(F)^2.
\end{aligned}$$

The general case follows by induction. □

Fact 7.3. *For any function $F\colon \mathbb{F}_2^n \to \{-1,1\}$ and for any $P = \mathsf{e}(p)$, where $p \in \mathrm{DEG}_d$, we have $\mathsf{U}_{d+1}(F\cdot P) = \mathsf{U}_{d+1}(F)$.*

Proof. Recall that

$$\mathsf{U}_{d+1}(F\cdot P) = \mathbb{E}_{\boldsymbol{h}\sim\mathcal{U}}\left[\mathbb{E}_{\boldsymbol{x},\boldsymbol{h}_1,\ldots,\boldsymbol{h}_{d+1}\sim\mathcal{U}}\left[F(\boldsymbol{x})F^{+\boldsymbol{h}_1}(\boldsymbol{x})\cdots F^{+\boldsymbol{h}_{d+1}}(\boldsymbol{x})P(\boldsymbol{x})P^{+\boldsymbol{h}_1}(\boldsymbol{x})\cdots P^{+\boldsymbol{h}_{d+1}}(\boldsymbol{x})\right]\right],$$

that is,

$$\mathsf{U}_{d+1}(F\cdot P) = \mathbb{E}_{\boldsymbol{h},\boldsymbol{h}_1,\ldots,\boldsymbol{h}_{d+1}\sim\mathcal{U}}\left[\prod_{S\subseteq[d+1]} F^{+\sum_{i\in S}\boldsymbol{h}_i}(\boldsymbol{x})P^{+\sum_{i\in S}\boldsymbol{h}_i}(\boldsymbol{x})\right].$$

Now p is a degree-d polynomial, and we saw last time that its "first discrete-derivative," $p(x)+p(x+h)$ has degree 1 less than $\deg(p)$. So $\sum_{S\subseteq[d+1]} P^{+\sum_{i\in S}\boldsymbol{h}_i}(\boldsymbol{x})$ is the $(d+1)$-st derivative, which equals 0 since $\deg(d) \leqslant p$. Therefore $P^{+\sum_{i\in S}\boldsymbol{h}_i}(\boldsymbol{x}) = P^0(\boldsymbol{x}) = 1$ for all $S \subseteq [d+1]$. Therefore

$$\begin{aligned}
U_{d+1}(F\cdot P) &= \mathop{\mathbb{E}}_{\boldsymbol{h},\boldsymbol{h}_1,\ldots,\boldsymbol{h}_{d+1}\sim\mathcal{U}}\left[\prod_{S\subseteq[d+1]} F^{+\sum_{i\in S}\boldsymbol{h}_i}(\boldsymbol{x})\cdot 1\right] \\
&= \mathbb{E}_{\boldsymbol{x},\boldsymbol{h}_1,\ldots,\boldsymbol{h}_{d+1}\sim\mathcal{U}}\left[F(\boldsymbol{x})F^{+\boldsymbol{h}_1}(\boldsymbol{x})\cdots F^{+\boldsymbol{h}_{d+1}}(\boldsymbol{x})\right] \\
&= \mathsf{U}_{d+1}(F).
\end{aligned}$$

□

We can now state and finish off our incomplete work from last time:

Lemma 7.4. *Let $F\colon \mathbb{F}_2^n \to \{-1,1\}$ be any function, and let $p\colon \mathbb{F}_2^n \to \{0,1\}$ be a degree-d polynomial, and let $P = \mathsf{e}(p)$. Then*

$$\mathrm{Cor}[F,P] \leqslant \mathsf{U}_{d+1}(F)^{\frac{1}{2^{d+1}}}.$$

Proof. Write $G(x) = F(x)\cdot P(x)$. Then we have that

$$\begin{aligned}
|\mathrm{Cor}[F,P]| &= |\mathbb{E}[F\cdot P]| = |\mathbb{E}[G]| \\
&= \sqrt{\mathsf{U}_1(G)} && \text{by Fact 7.1} \\
&\leqslant \mathsf{U}_2(G)^{1/4} \leqslant \mathsf{U}_3(G)^{1/8} \leqslant \ldots \leqslant \mathsf{U}_{d+1}(G)^{1/2^{d+1}} && \text{by Fact 7.2} \\
&= \mathsf{U}_{d+1}(F\cdot P)^{1/2^{d+1}} \\
&= \mathsf{U}_{d+1}(F)^{1/2^{d+1}} && \text{by Fact 7.3.}
\end{aligned}$$

□

§7.2 Basic tools for PRGs

We will now move into the second half of the course and start developing some of the notions that we will often return to as we think about derandomisation and pseudorandomness:

1. k-wise independent (or k-wise uniform) random variables (think of these as distributions over $\{0,1\}^n$).
2. ε-biased random variables.
3. k-wise independent ε-biased random variables.
4. Applications.

§7.2.1 k-wise independent random variables

Let us start by recalling the basic definition of true indepenence:

Definition 7.5 (Total independence)**.** *Let $\boldsymbol{X}_1,\ldots,\boldsymbol{X}_n$ be random variables, each supported on a finite set A. We say that the tuple $(\boldsymbol{X}_1,\ldots,\boldsymbol{X}_n)$ is* independent *if for all $a_1,\ldots,a_n \in A$, we have*

$$\Pr[\boldsymbol{X}_1 = a_1 \wedge \ldots \wedge \boldsymbol{X}_n = a_n] = \prod_{i=1}^{n} \Pr[\boldsymbol{X}_i = a_i].$$

The notion of k-wise independence is motivated by the following question: can we get away with a weaker notion of independence that is easier to achieve compared to total independence? The answer is yes:

Definition 7.6 (k-wise independence)**.** *Let $\boldsymbol{X}_1,\ldots,\boldsymbol{X}_n$ be random variables, each supported on a finite set A. We say that the tuple $(\boldsymbol{X}_1,\ldots,\boldsymbol{X}_n)$ is* k-wise independent *if for all $1 \leqslant i_1 < i_2 < \ldots < i_k \leqslant n$ and for all $a_1,\ldots,a_k \in A$, we have*

$$\Pr[\boldsymbol{X}_{i_1} = a_1 \wedge \ldots \wedge \boldsymbol{X}_{i_k} = a_k] = \prod_{j=1}^{k} \Pr[\boldsymbol{X}_{i_j} = a_j].$$

The most common version of k-wise independent random variables is when $k = 2$, in which case we call them *pairwise independent* random variables. A terminological annoyance here is that the term "k-wise uniform" is also used to refer to k-wise independent random variables, but this is just the special case where each x_i is uniform over the set $A = \{0, 1\}$.

Example 7.7. Suppose $\boldsymbol{X}_1 = \ldots = \boldsymbol{X}_n$ are all independent random variables, each uniform over $\{0, 1\}$. Then $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n)$ is 1-wise independent and 1-wise uniform.
Now let $X_1 = \mathcal{U}_1$ and $X_2 = \mathcal{U}_2$, where $\mathcal{U}_1, \mathcal{U}_2$ are independent uniform random variables over $\{0, 1\}$. Now let $\boldsymbol{X}_3 = \boldsymbol{X}_1 \oplus \boldsymbol{X}_2$, where $\oplus$ denotes addition modulo 2. Then $(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3)$ is 2-wise independent—for any a and b, we have that

$$\Pr[\boldsymbol{X}_{i_1} = a \wedge \boldsymbol{X}_{i_2} = b] = \frac{1}{4} = \Pr[\boldsymbol{X}_{i_1} = a] \Pr[\boldsymbol{X}_{i_2} = b].$$

Note that we only need to toss two coins to generate $\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3$.

So we might believe that if all we care about is k-wise independence, then we can achieve it with a small number of random bits, certainly smaller than n random bits. This is indeed the case:

Claim 7.8. *We can generate n pairwise uniform bits using seed length $\lceil \log_2(n+1) \rceil = k$, and this is optimal.*

Proof. Let $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k$ be independent uniform bits over $\{0, 1\}$. For each nonempty $S \subseteq [k]$, let

$$\boldsymbol{X}_S = \bigoplus_{i \in S} b_i.$$

Then $(\boldsymbol{X}_S)_{S \subseteq [k]}$ is a collection of $2^k - 1$ pairwise independent bits, because we are excluding the empty set and counting the number of bits in S. We want to show that these are pairwise uniform; that is, for every way of choosing one subset and a distinct subset, the probability that the first subset equals bit α and the second subset equals bit β is $\frac{1}{4}$.

Fix any two nonempty $S_1 \neq S_2 \subseteq [k]$. Consider any $(\alpha, \beta) \in \{0, 1\}^2$. Without loss of generality, take $S_1 \setminus S_2 \neq \emptyset$. Then we have that

$$\begin{aligned}\Pr_{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k}[\boldsymbol{X}_{S_1} = \alpha \wedge \boldsymbol{X}_{S_2} = \beta] &= \Pr_{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k}\left[\bigoplus_{i \in S_1} b_i = \alpha \wedge \bigoplus_{i \in S_2} b_i = \beta\right] \\ &= \Pr\left[\boldsymbol{X}_{S_1} = \alpha \mid \boldsymbol{X}_{S_2} = \beta\right] \cdot \Pr\left[\boldsymbol{X}_{S_2} = \beta\right].\end{aligned}$$

Now we claim that $\Pr[\boldsymbol{X}_{S_2} = \beta] = 1/2$. Why? We're asking for what fraction of the time what fraction of the time some nonempty parity gives us the value β after plugging in independent uniform bits. This is clearly $1/2$—fix outcomes for each $\boldsymbol{b}_i$ except for the last element j of S_2, and then one $\boldsymbol{b}_j$ outcome either causes $\boldsymbol{X}_{S_2}$ to be 0 and the other causes it to be 1. We can apply the same argument to $\Pr[\boldsymbol{X}_{S_1} = \alpha \mid \boldsymbol{X}_{S_2} = \beta]$. Let j' be an element of $S_1 \setminus S_2$. For $i \in S_2$, fix all b_i such that $\boldsymbol{X}_{S_2} = \beta$, and fix all the b_i other than $i = j'$. One setting of j' gives $\boldsymbol{X}_{S_1} = \alpha$, and the other setting gives $\boldsymbol{X}_{S_1} \neq \alpha$. Therefore $\Pr[\boldsymbol{X}_{S_1} = \alpha \mid \boldsymbol{X}_{S_2} = \beta] = 1/2$. Therefore

$$\Pr_{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k}[\boldsymbol{X}_{S_1} = \alpha \wedge \boldsymbol{X}_{S_2} = \beta] = \frac{1}{4}.$$

□

Application: Derandomising a simple randomised approximation algorithm for MAX-CUT**.** Let $G = (V, E)$ be a graph, and let $w\colon E \to \mathbb{R}_{\geqslant 0}$ be a weight function. The MAX-CUT problem is to find a partition of V into two sets $S, V \setminus S$ such that the total weight of edges crossing the partition is maximised (we will consider only unweighted nonsimple graphs, so we consider only the number of edges crossing the partition). This problem is famously NP-hard, even under the unique games conjecture. The following simple randomised algorithm gives a 1/2-approximation to MAX-CUT:

1. For each vertex $v \in V$, assign it to S with probability $1/2$ by tossing coins.
2. Output the cut $(S, V \setminus S)$.

Let the expected number of edges crossing from S to $V \setminus S$ be $\mathbb{E}\left[|\delta(S)|\right]$. Then

$$\mathbb{E}\left[|\delta(S)|\right] = \frac{1}{2}\sum_{(u,v)\in E} \Pr[u \in S \wedge v \in V \setminus S] = \frac{1}{2}|E| \leqslant \frac{1}{2}\mathrm{OPT},$$

where OPT is the maximum number of edges that can be cut. Therefore the algorithm is a 1/2-approximation algorithm. We can derandomise this algorithm using pairwise independent random variables. This algorithm tossed $n = |V|$ coins, and since, in the analysis of the algorithm we didn't need independent uniform n-bit strings but only cared about pairwise events (i.e. for each pair of vertices, whether each lies on either end of the cut) we can replace these with pairwise independent random bits and the analysis still works. So we could enumerate over all $2^{\lceil \log_2(n+1)\rceil} = O(n)$ possible strings in the support of the pairwise uniform distribution, and use the one that maximises the number of edges cut. This gives us a poly-time deterministic 1/2-approximation algorithm for MAX-CUT.

Let us now stop obsessing about pairwise uniform random variables and focus more broadly on k-wise uniform random variables, where $k > 2$. Let $|\mathbb{F}|$ be a finite field of size n, where n is a prime power (say 2^ℓ for some ℓ). We give the following construction for an n-tuple of k-wise uniform random variables over $\mathbb{F}$. Let $\boldsymbol{c}_0, \ldots, \boldsymbol{c}_{k-1}$ be independent uniform random variables sampled from $\mathbb{F}$ with $k \cdot \log n$ bits of true randomness (i.e. the seed length). View these as coefficients of a univariate polynomial over $\mathbb{F}$; that is, for each element $\alpha \in \mathbb{F}$, let

$$\boldsymbol{X}_\alpha = \sum_{i=0}^{k-1} \boldsymbol{c}_i \cdot \alpha^i. = p_{\boldsymbol{c}}(\alpha).$$

where $p_{\boldsymbol{c}}$ is a degree-$(k-1)$ polynomial over $\mathbb{F}$. We have just defined an n-tuple of elements over the field $\mathbb{F}$, and it remains to argue that this field is k-wise uniform, that is, the probability for any k-tuple outcome is the product of the probabilities of the individual outcomes. First we focus on a particular individual outcome: for any fixed $\alpha \in \mathbb{F}$, the random variable $\boldsymbol{X}_\alpha$ is uniform over $\mathbb{F}$, because a fixed c_0 makes it so regardless of $\alpha, c_1, \ldots, c_{k-1}$. With this fact, we can argue for k-tuples of randomness, and we will indeed get k-wise independence by Lagrange interpolation, which says that for any desired vector of outcomes $(a_1, \ldots, a_k) \in \mathbb{F}^k$, and for any distinct $\alpha_1, \ldots, \alpha_k \in \mathbb{F}$, there is a unique $c = (c_0, \ldots, c_{k-1})$ giving a unique degree-$(k-1)$ polynomial p such that $X_{\alpha_i} = p(\alpha_i) = a_i$ for all i in $[k]$, defined as

$$X_\alpha = p(\alpha) = \sum_{i=1}^{k} a_i \cdot \frac{\prod_{j\neq i}(\alpha - \alpha_j)}{\prod_{j\neq i}(\alpha_i - \alpha_j)}.$$

This is because the Lagrange basis polynomials are linearly independent, and so the polynomial p is unique. Returning back to randomness, we have that a uniform distribution over $(\boldsymbol{c}_0, \ldots, \boldsymbol{c}_{k-1})$ induces a uniform distribution over $(\alpha_1, \ldots, \alpha_k)$, and so

$$\Pr[\boldsymbol{X}_{\alpha_1} = a_1 \wedge \ldots \wedge \boldsymbol{X}_{\alpha_k} = a_k] = \frac{1}{|\mathbb{F}|^k} = \prod_{i=1}^{k} \Pr[\boldsymbol{X}_{\alpha_i} = a_i].$$

Therefore we have a k-wise uniform distribution over $\mathbb{F}$, the field of size n.

Application: Derandomising a simple randomised approximation algorithm for Max-3-SAT**.** Let $\phi = C_1 \wedge \ldots \wedge C_m$ be a 3-CNF formula with m clauses $C_i = \ell_{i_1} \vee \ell_{i_2} \vee \ell_{i_3}$, where each ℓ_{i_j} is a literal. The Max-3-SAT problem is to find an assignment to the variables that satisfies the maximum number of clauses. This is also an NP-hard problem, and there is a a simple 7/8-approximation to Max-3-SAT:

1. Pick a uniformly random assignment to the variables.
2. Output the assignment.

Why is this a 7/8-approximation? Let OPT be the maximum number of clauses that can be satisfied. Then the expected number of clauses satisfied by the random assignment is

$$\mathbb{E}[\text{SAT}] = \sum_{i=1}^{m} \Pr[C_i \text{ is satisfied}] = \frac{7}{8}m \geqslant \frac{7}{8}\text{OPT}.$$

Parsimonous to our previous application, we don't need a truly random assignment for the three variables, but only to have drawn a 3-tuple of k-wise uniform random variables over $\mathbb{F}_2$. We can then enumerate over all strings a in the support of our favourite 3-wise independent distribution over $\{0,1\}^n$, and the best outcome maximises the number of clauses satisfied, and satisfies at least 7/8 of the clauses. This gives us a poly-time (since for $k = 3$ we get runtime $2^{3 \log n} = \text{poly}(n)$) deterministic 7/8-approximation algorithm for Max-3-SAT.

§7.2.2 A PRG-type application: juntas and bounded-depth decision trees

A k-junta is an n-variable function that really just depends on k of its input variables.

Definition 7.9 (Juntas)**.** *A function $f\colon \{0,1\}^n \to \{0,1\}$ is a* k-junta *if there exists a set $S \subseteq [n]$ with $|S| \leqslant k$ such that for all $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n$, we have*

$$f(\boldsymbol{x}) = f(\boldsymbol{y}) \text{ if and only if } \boldsymbol{x}|_S = \boldsymbol{y}|_S,$$

where $\boldsymbol{x}|_S$ denotes the restriction of $\boldsymbol{x}$ to the coordinates in S.

We will write $\mathcal{J}_k$ to denote the class of all k-juntas, and $\mathcal{J} = \bigcup_{k=0}^{n} \mathcal{J}_k$ to denote the set of all juntas. It is immediate from the definition of k-wise uniformity that if f is a k-junta, then f is perfectly fooled (0-fooled) by k-wise independent random variables with seed length $k \cdot \log n$.

A notion that is slightly less immediate but barely any less immediate is that of decision trees:

Definition 7.10 (Decision trees)**.** *Define* DT_k *to be the set of all functions* $f\colon \{0,1\}^n \to \{0,1\}$ *that can be computed by a decision tree of depth at most* k*.*

Notice that this is a richer class than $\mathcal{J}_k$; in particular, we can show that $\mathcal{J}_k \subsetneq \mathrm{DT}_k$ (as we can construct, eg. a decision tree of depth 3 that depends on more than three variables). Any decision tree is a $\approx k \cdot 2^k$-junta, but doesn't have to depend on only k variables. We can show that if we perfectly fool k-juntas, then we also perfectly fool decision trees of depth k.

Claim 7.11. *If* $\boldsymbol{X}$ 0*-fools* $\mathcal{J}_k$*, then* $\boldsymbol{X}$ 0*-fools* DT_k*.*

Proof. Let $f \in \mathrm{DT}_k$. Then $f = \sum_{\ell \in L} f_\ell$, where L is the set of 1-leaves of the decision tree, and each f_ℓ is a k-junta (in fact it is a size-k conjunction of literals). Therefore

$$\begin{aligned}
\mathbb{E}[f(\boldsymbol{X})] &= \mathbb{E}\left[\sum_{\ell \in L} f_\ell(\boldsymbol{X})\right] = \sum_{\ell \in L} \mathbb{E}[f_\ell(\boldsymbol{X})] \\
&= \sum_{\ell \in L} \mathbb{E}[f_\ell(\mathcal{U})] && \text{since } \boldsymbol{X} \text{ 0-fools } \mathcal{J}_k \\
&= \mathbb{E}\left[\sum_{\ell \in L} f_\ell(\mathcal{U})\right] \\
&= \mathbb{E}[f(\mathcal{U})],
\end{aligned}$$

which is what we wanted, since the difference between $\mathbb{E}[f(\boldsymbol{X})]$ and $\mathbb{E}[f(\mathcal{U})]$ is 0 (here $\mathcal{U}$ is the uniform distribution over $\{0,1\}^n$). □

We can generalise this slightly to get a triangle inequality:

Lemma 7.12 (Triangle inequality for PRG errors)**.** *Let* $f_1, \dots, f_t\colon \{0,1\}^n \to \mathbb{R}$*. Let* $\lambda_0, \dots, \lambda_t \in \mathbb{R}$*, and consider the linear combination*

$$f(x) = \lambda_0 + \sum_{i=1}^{t} \lambda_i f_i(x).$$

If $\boldsymbol{X}$ ε_i*-fools each* f_i *for all* $i \in [t]$*, then* $\boldsymbol{X}$ ε*-fools* f*, where* $\varepsilon = \sum_{i=1}^{t} |\lambda_i| \cdot \varepsilon_i$*.*

Proof. We have that

$$\begin{aligned}
|\mathbb{E}[f(\boldsymbol{X})] - \mathbb{E}[f(\mathcal{U})]| &= \left|\sum_{i=1}^{t} \lambda_i \left(\mathbb{E}[f_i(\boldsymbol{X})] - \mathbb{E}[f_i(\mathcal{U})]\right)\right| \\
&\leqslant \sum_{i=1}^{t} |\lambda_i| \cdot |\mathbb{E}[f_i(\boldsymbol{X})]| - \sum_{i=1}^{t} |\lambda_i| \cdot |\mathbb{E}[f_i(\mathcal{U})]| \\
&\leqslant \sum_{i=1}^{t} |\lambda_i| \cdot \varepsilon_i.
\end{aligned}$$

□

§7.2.3 ε-biased distributions over $\{0,1\}^n$

The elevator pitch that an ε-biased distribution over $\{0,1\}^n$ will give us is that it is a distribution that fools parity functions.

Definition 7.13 (Parity functions)**.** *The class of all parity functions over $\{0,1\}^n$, written PARITY_n is the set of all functions $p_S\colon \{0,1\}^n \to \{0,1\}$, where $S \subseteq [n]$, and*

$$p_S(x) = \sum_{i \in S} x_i \pmod 2 = \bigoplus_{i \in S} x_i.$$

Sometimes it will be more convenient to think in terms of the *character function*, which is the function $\chi_S\colon \{0,1\}^n \to \{-1,1\}$, where

$$\chi_S(x) = (-1)^{p_S(x)} = \mathsf{e}\left(p_S(x)\right).$$

We can now define the subject of this section:

Definition 7.14. *A random variable $\boldsymbol{X} = (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n)$ over $\{0,1\}^n$ is ε-*biased *if it $(\varepsilon/2)$-fools the class PARITY_n. That is, it ε-fools the class of all character functions χ_S:*

$$\left|\mathbb{E}[\chi_S(\boldsymbol{X})] - \mathbb{E}[\chi_S(\mathcal{U})]\right| \leqslant \frac{\varepsilon}{2},$$

where $\mathbb{E}[\chi_S(\mathcal{U})] = 0$ when $S \neq \emptyset$ and $\mathbb{E}[\chi_S(\mathcal{U})] = 1$ when $S = \emptyset$. In particular, for $S \neq \emptyset$, we have that

$$\frac{1}{2} - \frac{\varepsilon}{2} \leqslant \Pr[p_S(\boldsymbol{X})] \leqslant \frac{1}{2} + \frac{\varepsilon}{2}.$$

Although they will not be enough by themselves, we will use ε-biased distributions to fool the class DEG_d of all $\mathbb{F}_2$-polynomials of degree at most d, as long as d is not too large.

Construction of ε-biased random variables The following construction is due to [AGHP92]. Let $b_{ij}\colon \mathbb{F}_{2^\ell} \to (\mathbb{F}_2)^\ell$ be a be a linear bijection so that for all $i, j \in [\ell]$, we have that $b_{ij}(x+y) = b_{ij}(x) + b_{ij}(y)$. (Here $\mathbb{F}_{2^\ell}$ is the finite field with elements as degree $\leqslant \ell - 1$ polynomials over $\mathbb{F}_2$, modulo some fixed irreducible polynomial of degree ℓ, and $(\mathbb{F}_2)^\ell$ is the vector space of dimension ℓ over $\mathbb{F}_2$.) Our standard construction of elements of $\mathbb{F}_{2^\ell}$ gives us such a bijection: we add polynomials by adding their coefficients modulo 2. So we have such a bijection. Let $\ell = \log\left(\frac{n}{\varepsilon}\right)$. Then the ε-biased generator $G\colon \mathbb{F}_{2^\ell}^2 \to \{0,1\}^n$ is defined as follows: $G(x,y) = (r_0, \ldots, r_{n-1})$, where

$$r_i = \left\langle b_{ij}(y), b_{ij}(x^i) \right\rangle \pmod 2,$$

where $\langle \cdot, \cdot \rangle$ denotes the standard inner product[5]. over $\mathbb{F}_2^\ell$; note that both $b_{ij}(y)$ and $b_{ij}(x^i)$ are elements of $(\mathbb{F}_2)^\ell$, and further that the seed length of G is $2\ell = 2\log\left(\frac{n}{\varepsilon}\right) = \log\left(\frac{n^2}{\varepsilon^2}\right)$, since we need to map pairs of elements of $\mathbb{F}_{2^\ell}$ to elements of $\{0,1\}^n$. We can show that G is ε-biased by means of the following lemma:

[5] That is, $\langle (a_1, \ldots, a_\ell), (b_1, \ldots, b_\ell) \rangle = \sum_{i=1}^{\ell} a_i b_i \pmod 2$.

Lemma 7.15. *This G is an ε-biased generator: for uniform $\boldsymbol{u} \sim \{0,1\}^{2\log\left(\frac{n}{\varepsilon}\right)}$, we have that $G(\boldsymbol{u})$ is an ε-biased random variable.*

Proof. We want to show that we can fool all parities, that is, for all nonzero $\alpha \in \{0,1\}^n$ indexing the parities, we have that

$$\left| \Pr_{\boldsymbol{r} \sim G(\mathcal{U})} \left[\sum_{i=0}^{n-1} \alpha_i \boldsymbol{r}_i \equiv 1 \pmod 2 \right] - \frac{1}{2} \right| \leqslant \frac{\varepsilon}{2}.$$

By the definition of G, we get

$$\begin{aligned}
\Pr_{\boldsymbol{r} \sim G(\mathcal{U})} \left[\sum_{i=0}^{n-1} \alpha_i \boldsymbol{r}_i \equiv 1 \pmod 2 \right] &= \Pr_{\boldsymbol{x},\boldsymbol{y} \sim \mathbb{F}_{2^\ell}} \left[\sum_{i=0}^{n-1} \alpha_i \left\langle b_{ij}(\boldsymbol{y}), b_{ij}(\boldsymbol{x}^i) \right\rangle \equiv 1 \pmod 2 \right] \\
&= \Pr_{\boldsymbol{x},\boldsymbol{y} \sim \mathbb{F}_{2^\ell}} \left[\left\langle b_{ij}(\boldsymbol{y}), b_{ij} \left(\sum_{i=0}^{n-1} \alpha_i \boldsymbol{x}^i \right) \right\rangle \equiv 1 \pmod 2 \right] \\
&= \Pr_{\boldsymbol{x},\boldsymbol{y} \sim \mathbb{F}_{2^\ell}} \left[\left\langle b_{ij}(\boldsymbol{y}), b_{ij} \left(p_\alpha(\boldsymbol{x}) \right) \right\rangle \equiv 1 \pmod 2 \right],
\end{aligned}$$

where $p_\alpha(x) = \sum_{i=0}^{n-1} \alpha_i x^i$ is the $\mathbb{F}_{2^\ell}$-polynomial of degree-$(n-1)$, and the second equality above follows from the linearity of the bijection b_{ij}. Now define the event E as

$$E = \left\{ \left\langle b_{ij}(\boldsymbol{y}), b_{ij} \left(p_\alpha(\boldsymbol{x}) \right) \right\rangle \equiv 1 \pmod 2 \right\}.$$

Now we do casework on x; either $p_\alpha(x) = 0$ or $p_\alpha(x) \neq 0$. If $p_\alpha(x) = 0$, then

$$\Pr_{\boldsymbol{y} \sim \mathbb{F}_{2^\ell}} \left[\left\langle b_{ij}(\boldsymbol{y}), b_{ij} \left(p_\alpha(\boldsymbol{x}) \right) \right\rangle \equiv 1 \pmod 2 \right] = 0,$$

by linearity and since the inner product is always even. If $p_\alpha(x) \neq 0$, then

$$\Pr_{\boldsymbol{y} \sim \mathbb{F}_{2^\ell}} \left[\left\langle b_{ij}(\boldsymbol{y}), b_{ij} \left(p_\alpha(\boldsymbol{x}) \right) \right\rangle \equiv 1 \pmod 2 \right] = \frac{1}{2},$$

since the inner product (the parity) is uniformly distributed over $\mathbb{F}_2$. Therefore

$$\begin{aligned}
\Pr_{\boldsymbol{r} \sim G(\mathcal{U})} \left[\sum_{i=0}^{n-1} \alpha_i \boldsymbol{r}_i \equiv 1 \pmod 2 \right] &= \Pr_{\boldsymbol{x},\boldsymbol{y} \sim \mathbb{F}_{2^\ell}} [E] \\
&= \Pr_{\boldsymbol{y} \sim \mathbb{F}_{2^\ell}} [E \mid p_\alpha(\boldsymbol{x}) = 0] \Pr [p_\alpha(\boldsymbol{x}) = 0] \\
&\qquad + \Pr_{\boldsymbol{y} \sim \mathbb{F}_{2^\ell}} [E \mid p_\alpha(\boldsymbol{x}) \neq 0] \Pr [p_\alpha(\boldsymbol{x}) \neq 0] \\
&\qquad = 0 \cdot \Pr [p_\alpha(\boldsymbol{x}) = 0] + \frac{1}{2} \cdot \Pr [p_\alpha(\boldsymbol{x}) \neq 0] \leqslant \frac{1}{2},
\end{aligned}$$

establishing the upper bound. For the lower bound, we recall that $p_\alpha(x)$ is a degree-$(n-1)$ polynomial over $\mathbb{F}_{2^\ell}$; remember that $2^\ell = n/\varepsilon$, also, the degree-$(n-1)$ polynomial p_α has at most $n-1$ roots, and therefore

$$\Pr_{\boldsymbol{x} \sim \mathbb{F}_{2^\ell}} [p_\alpha(\boldsymbol{x}) \neq 0] \geqslant 1 - \frac{n-1}{2^\ell} = 1 - \frac{n-1}{n/\varepsilon} \geqslant 1 - \varepsilon.$$

So we get that

$$\Pr_{\boldsymbol{r}\sim G(\mathcal{U})}\left[\sum_{i=0}^{n-1}\alpha_i \boldsymbol{r}_i \equiv 1 \pmod 2\right] \geqslant \frac{1}{2} - \frac{\varepsilon}{2},$$

and we are done. □

k-wise independent distributions over $\mathbb{F}_2^n$ and ε-biased distributions over $\{0,1\}^n$ are closely related to linear codes over $\mathbb{F}_2^n$. (A code is a collection of vectors in a vector space that are all at least a certain distance apart from each other, and a linear code is a code that resides in a linear subspace of the vector space.) In particular, the k-wise-ness of k-wise independent distributions is related to the minimum distance of the code, and the ε-biasedness of ε-biased distributions is related to the balancedness of the code. There is a coding-theoretic motivation to do better for these ε-biased distributions just seen, than our $2\log n/\varepsilon$ seed length. It is conceivable that we could get explicit $(\log n + 2\log 1/\varepsilon - \log\log 1/\varepsilon)$-seed-length ε-bias might exist. The work of Amnom Ta-Shma [TS17] gives a construction of a generator with seed length $(\log n + 2\log(1/\varepsilon) + \widetilde{O}(\log^{2/3}(1/\varepsilon)))$, and might be worth looking into for a project topic.

§7.2.4 The best of both worlds: k-wise independent ε-biased random variables

Definition 7.16. *A random variable $\boldsymbol{X} = (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n)$ over $\{0,1\}^n$ is* k-wise ε-biased *if it fools all χ_S with $|S| \leqslant k$.*

So of course the k-wise independent distribution 0-fools parities of size at most k, and the ε-biased distribution 0-fools all parities. Therefore, we can expect that k-wise ε-biased distributions will be cheaper to construct than either k-wise independent or ε-biased distributions. Indeed this is true:

Lemma 7.17 (Naor and Naor [NN90])**.** *There exists an explicit k-wise ε-biased generator $\boldsymbol{X} = (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n) = G(\mathcal{U})$ with seed length $O\left(\log k + \log\frac{1}{\varepsilon} + \log\log n\right)$.*

Proof. Let $G\colon \{0,1\}^s \to \{0,1\}^n$ be a k-wise uniform generator with seed length $s = O(k\log n)$ be a linear transformation viewed as a mapping $(\mathbb{F}_2)^s \mapsto (\mathbb{F}_2)^n$. (Our k-wise uniform generator/distribution is such.)

Let $\mathcal{Y}$ be an ε-biased distribution over $\{0,1\}^s$. Overall, $\mathcal{X} = G(\mathcal{Y})$ outputs n bits, and its seed length is $2\log(\frac{s}{\varepsilon}) = O\left(\log k + \log\frac{1}{\varepsilon} + \log\log n\right)$. We want to show that $G(\mathcal{Y})$ ε-fools parities of size at most k. Let $S \subseteq [n]$ with $|S| \leqslant k$. Pick $p_S(x) = \sum_{i\in S} x_i$, where $x \in \{0,1\}^n$; this is the parity we intend to fool. Let $M \in \mathbb{F}_2^{n\times s}$ be the matrix that represents the linear transformation G:

$$M = \begin{bmatrix} \text{———} & M_1 & \text{———} \\ \text{———} & M_2 & \text{———} \\ & \vdots & \\ \text{———} & M_n & \text{———} \end{bmatrix}.$$

Here each row $M_i \in (\mathbb{F}_2)^s$. More explicitly,

$$G(\mathcal{Y}) = (\langle M_1, \mathcal{Y}\rangle, \ldots, \langle M_n, \mathcal{Y}\rangle) \in \mathbb{F}_2^n.$$

For $y \in \mathbb{F}_2^s$, we have that

$$\begin{aligned} p_S\left(G(\mathcal{Y})\right) &= \sum_{i \in S} \langle M_i, \mathcal{Y} \rangle = \sum_{i \in S} \sum_{j=1}^{s} M_{ij} \mathcal{Y}_j \\ &= \sum_{j=1}^{s} \left(\sum_{i \in S} M_{ij} \right) \mathcal{Y}_j. \end{aligned}$$

This is some PARITY over $\mathcal{Y}_1, \ldots, \mathcal{Y}_s$, and so we can use the ε-biasedness of $\mathcal{Y}$ to get that

$$\left| \mathbb{E}\left[p_S\left(G(\mathcal{Y})\right)\right] - \mathbb{E}\left[p_S\left(G\left(\mathcal{U}\right)\right)\right] \right| \leqslant \frac{\varepsilon}{2}.$$

Since p_S is a k-junta and G is k-wise uniform, we get that

$$\mathbb{E}\left[p_S\left(G\left(\mathcal{U}\right)\right)\right] = \mathbb{E}\left[p_S\left(\mathcal{U}\right)\right],$$

and so

$$\left| \mathbb{E}\left[p_S\left(G(\mathcal{Y})\right)\right] - \mathbb{E}\left[p_S\left(\mathcal{U}\right)\right] \right| \leqslant \frac{\varepsilon}{2},$$

that is, $\mathcal{Y}$ $(\varepsilon/2)$-fools p_S. Therefore $G(\mathcal{Y})$ is k-wise ε-biased. □

Next time we will see more applications of the tools we saw today (both easy ones and non-easy ones), as well as some Fourier analysis of Boolean functions.

§8 Lecture 08—05th March, 2024

Last time. In the previous class, we finished the first half of the course and started talking more about pseudorandom generators.

- First we finished the proof of the missing piece from the second correlation bound:

  Let $F\colon \mathbb{F}_2^n \to \{-1,1\}$ be any function, and let $p\colon \mathbb{F}_2^n \to \{0,1\}$ be a degree-d polynomial, and let $P = \mathsf{e}(p)$. Then

  $$\mathrm{Cor}[F,P] \leqslant \mathsf{U}_{d+1}(F)^{\frac{1}{2^{d+1}}}.$$

- Then we started the basic tools for PRGs:
  - k-wise independence of random variables and why they exist.
  - ε-biased random variables.
  - k-wise independent ε-biased random variables.

Today. We will:

- See some basic Fourier analysis over the Boolean hypercube; some applications: fooling size-s decision trees, and fooling k-juntas better.
- Sandwiching approximators and fooling.
- Viola's theorem: the sum of d-many ε-biased random variables fools degree-d polynomials.

---

§8.1 Interlude: some basic Fourier analysis over the Boolean hypercube

Many of the results we will discuss now can be found in greater detail in the popular book by Ryan O'Donnell [O'D14].

Now we will be concerned with functions like $f\colon \{-1,1\}^n \to \mathbb{R}$ or $f\colon \{0,1\}^n \to \mathbb{R}$. The collection of all such functions form a 2^n-dimensional vector space assigning one dimension for each x, with inner product

$$\langle f, g\rangle = \mathbb{E}_{\boldsymbol{u}\sim\mathcal{U}_n}[f(\boldsymbol{u})g(\boldsymbol{u})],$$

in particular, $\|f\| = \sqrt{\langle f, f\rangle}$. We will also write $\mathbb{E}[f] = \mathbb{E}_{\boldsymbol{u}\sim\mathcal{U}_n}[f(\boldsymbol{u})]$. A natural basis for this space we are considering consists of all 2^n parity functions:

Claim 8.1. *The set $(\chi_S)_{S\subseteq[n]}$ of all 2^n character functions $\chi_S\colon \{-1,1\}^n \to \{-1,1\}$, defined by $\chi_S(u) = (-1)^{\sum_{i\in S} u_i}$, forms an orthonormal basis for the space of all functions $f\colon \{-1,1\}^n \to \mathbb{R}$.*

Proof. We need to show that $\langle \chi_S, \chi_T\rangle = 0$ if $S \neq T$ and $\langle \chi_S, \chi_S\rangle = 1$ for all $S \subseteq [n]$. In the general case,

$$\begin{aligned}\langle \chi_S, \chi_T\rangle &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}_n}[\chi_S(\boldsymbol{x})\chi_T(\boldsymbol{x})] = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}_n}\left[(-1)^{\sum_{i\in S} x_i}(-1)^{\sum_{j\in T} x_j}\right]\\ &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}_n}[(-1)^{\sum_{i\in S\,\triangle\, T} x_i}] = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}_n}[(-1)^{|S\,\triangle\, T|}]\\ &= \mathbb{E}\left[\chi_{S\,\triangle\, T}(\boldsymbol{x})\right].\end{aligned}$$

So if $S \neq T$, then $|S \triangle T| = n$, and so $\langle \chi_S, \chi_T\rangle = 0$. If $S = T$, then $|S \triangle T| = 0$, and so $\langle \chi_S, \chi_T\rangle = 1$. □

So any function $f\colon \{0,1\}^n \to \mathbb{R}$ has a unique representation as a linear combination of the basis elements χ_S. Write $\widehat{f}(S)$ to denote the coefficient of χ_S in this representation, i.e. the *Fourier coefficients* of f. Then

$$f = \sum_{S\subseteq[n]} \widehat{f}(S)\chi_S,$$

and in particular,

$$\begin{aligned}\langle f, \chi_S\rangle &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}_n}[f(\boldsymbol{x})\chi_S(\boldsymbol{x})] = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}_n}\left[\left(\sum_{T\subseteq[n]} \widehat{f}(T)\chi_T(\boldsymbol{x})\right)\chi_S(\boldsymbol{x})\right]\\ &= \sum_{T\subseteq[n]} \widehat{f}(T)\cdot\mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}_n}[\chi_T(\boldsymbol{x})\chi_S(\boldsymbol{x})]\\ &= \widehat{f}(S).\end{aligned}$$

Thus $\widehat{f}(S)$ is a measure of the correlation of f and χ_S. In particular,

$$\widehat{f}(\emptyset) = \mathbb{E}\left[f(x)\cdot\chi_\emptyset(x)\right] = \mathbb{E}[f(x)].$$

Now take $f\colon \{-1,1\} \to \mathbb{R}$. Then

$$f(x) = \sum_{S\subseteq[n]} \widehat{f}(S)\cdot\prod_{i\in S} x_i,$$

and the Fourier representation is the exact same as the representation of f as a multilinear polynomial in the variables $x_1, \ldots, x_n$.

There is a useful statement about the inner product of two functions in boolean Fourier analysis:

Proposition 8.2 (Plancherel's identity)**.** *For any two functions $f, g\colon \{0,1\}^n \to \mathbb{R}$, we have that $\langle f, g\rangle = \sum_S \widehat{f}(S)\widehat{g}(S)$.*

Proof. Write

$$\begin{aligned}\langle f, g\rangle &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}_n}[f(\boldsymbol{x})g(\boldsymbol{x})] \\ &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}_n}\left[\left(\sum_S \widehat{f}(S)\chi_S(\boldsymbol{x})\right)\left(\sum_T \widehat{g}(T)\chi_T(\boldsymbol{x})\right)\right] \\ &= \sum_{S,T} \widehat{f}(S)\cdot\widehat{g}(T)\cdot\mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}_n}[\chi_S(\boldsymbol{x})\chi_T(\boldsymbol{x})] \\ &= \sum_{S,T} \widehat{f}(S)\cdot\widehat{g}(T)\cdot\delta_{S,T} = \sum_S \widehat{f}(S)\widehat{g}(S),\end{aligned}$$

since $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}_n}[\chi_S(\boldsymbol{x})\chi_T(\boldsymbol{x})] = \delta_{S,T}$. □

The following corollary is immediate:

Corollary 8.3 (Parseval's identity)**.** *For any function $f\colon \{0,1\}^n \to \mathbb{R}$, we have $\|f\|^2 = \sum_S \widehat{f}(S)^2$.*

Proof. This follows from Plancherel's identity by setting $f = g$:

$$\|f\|^2 = \langle f, f\rangle = \mathbb{E}_{x\sim\mathcal{U}}[f(\boldsymbol{x})^2] = \sum_S \widehat{f}(S)^2.$$

Indeed if $f\colon \{0,1\}^n \to \{-1,1\}$, then $\|f\|^2 = \mathbb{E}[f^2] = 1 = \sum_S \widehat{f}(S)^2$. □

Fourier representations will be useful for us as we continue. In fact, we can show that (and this is left as an exercise) for all degree-1 $\mathbb{F}_2$-polynomials $p(x)$, we have

$$\Pr_{\boldsymbol{u}\sim\mathcal{U}_n}[\mathrm{IP}(\boldsymbol{u}) = p(\boldsymbol{u})] = \frac{1}{2} \pm \frac{1}{2^{n/2}},$$

by writing down the Fourier representation of $\mathrm{IP}\colon \{0,1\}^n \to \{\pm 1\}$ and using Parseval's identity.

We end our interlude with the following definition:

Definition 8.4 (Fourier L_1-norm)**.** *The* Fourier L_1-norm *of a function $f\colon \{0,1\}^n \to \mathbb{R}$ is defined as*

$$L_1(f) = \sum_{S\subseteq[n], S\neq\emptyset} |\widehat{f}(S)|.$$

The following fact is immediate:

Fact 8.5. *If $f\colon \{0,1\}^n \to [-1,1]$, then we have that $L_1(f) \leqslant 2^{n/2}$.*

Proof. By Cauchy-Schwarz,

$$\begin{aligned} L_1(f) &= \sum_{S\subseteq[n]} |\widehat{f}(S)| \leqslant \sqrt{2^n \cdot \sum_{S\subseteq[n]} \widehat{f}(S)^2} \\ &= \sqrt{2^n \cdot \mathbb{E}[f(\boldsymbol{x})^2]} \leqslant \sqrt{2^n}, \end{aligned}$$

where the last inequality follows from the fact that $f(\boldsymbol{x})^2 \leqslant 1$ for all $\boldsymbol{x}$. □

§8.2 Applications of the basic tools: fooling juntas and decision trees better

Recall the "triangle inequality for PRGs" from last time:

Lemma 8.6 (Triangle inequality for PRG errors)**.** *Let $f_1, \dots, f_t \colon \{0,1\}^n \to \mathbb{R}$. Let $\lambda_0, \dots, \lambda_t \in \mathbb{R}$, and consider the linear combination*

$$f(x) = \lambda_0 + \sum_{i=1}^{t} \lambda_i f_i(x).$$

If $\boldsymbol{X}$ ε_i-fools each f_i for all $i \in [t]$, then $\boldsymbol{X}$ ε-fools f, where $\varepsilon = \sum_{i=1}^t |\lambda_i| \cdot \varepsilon_i$.

Now we are thinking of Boolean functions as linear combinations of the basis functions χ_S, and we discussed the ε-biased PRG which fools character functions last time, so we have a tailor-made situation for this inequality to help us do even more. The following is a corollary of Lemma 8.6:

Corollary 8.7. *If $\boldsymbol{X}$ is a δ-biased random variable over $\{0,1\}^n$, then $\boldsymbol{X}$ fools all functions $f \colon \{0,1\}^n \to \mathbb{R}$ with error $\delta \cdot L_1(f)$.*

One application of this corollary is the following, which is also a required exercise: if $f \colon \{0,1\}^n \to \{0,1\}$ is a conjunction of literals over distinct variables, then $L_1(f) = 1$. Last time, we 0-fooled depth-d decision trees using d-wise independence with seed length $O(d \log n)$. Now we will ε-fool the (broader) class of size-s decision trees using δ-biased random variables.

Claim 8.8. *If $\boldsymbol{X}$ is a δ-biased random variable over $\{0,1\}^n$, then $\boldsymbol{X}$ fools any size-s decision tree with error $\delta \cdot s$. Consequently we get an ε-PRG for decision trees of size $\leqslant s$ with seed length $O\left(\log s + \log n + \log \frac{1}{\varepsilon}\right)$.*

Proof. Let T be a size-s decision tree, let L be the set of its leaves that are labelled 1, and let f_ℓ be the conjunction corresponding to a given 1-labelled leaf $\ell \in L$. Then $f = \sum_{\ell \in L} f_\ell$, and so by the triangle inequality,

$$L_1(f) \leqslant \sum_{\ell \in L} L_1(f_\ell) \leqslant |L| \leqslant s,$$

so we are done, by Corollary 8.7 and the homework exercise. □

The significance of this claim is that we can now fool the richer class of decision trees of polynomial size as opposed to those of logarithmic depth (which we saw last time), by using δ-biased random variables, up to some error—we have done better.

A side remark is the following. Note that given a size-s decision tree T, it is easy to exactly compute in polynomial time the probability that a uniform random input satisfies the tree; in fact, it is

$$\Pr_{\boldsymbol{u}\sim\mathcal{U}_n}[T(\boldsymbol{u})=1]=\sum_{\text{1-leaf }\ell}2^{-\text{depth of }\ell},$$

so the PRG we constructed doesn't necessarily need to see the tree.

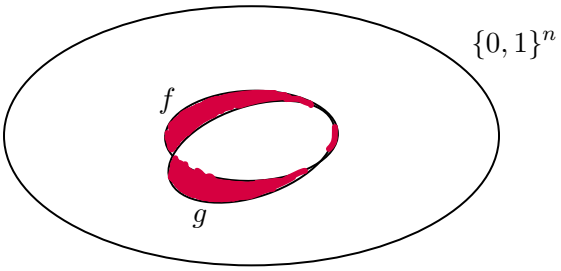
Let us fool juntas better than we did last time, where we used k-wise ε-biased random variables to fool k-juntas with seed length $O(k\log n)$.

Claim 8.9. *If $\boldsymbol{X}$ is k-wise ε-biased random variable[6] over $\{0,1\}^n$, then $\boldsymbol{X}$ fools $\{-1,1\}$-valued k-juntas with error $\delta\cdot 2^{k/2}=\varepsilon$ with seed length $O\left(k+\log\frac{1}{\varepsilon}+\log\log n\right)$.*

Proof. The function f is a k-junta, so $f(x)=g(x_{i_1},\dots,x_{i_k})$ for some $i_1,\dots,i_k\in[n]$ and some $g\colon\{0,1\}^k\to\{-1,1\}$. Then since $L_1(g)\leqslant 2^{k/2}$, the random variable $\boldsymbol{X}$ $\delta\cdot 2^{k/2}$-fools g, and therefore f. □

§8.3 Sandwiching and approximation

Suppose we want to fool a Boolean function $f\colon\{0,1\}^n\to\{0,1\}$ to error ε, and we have a function $g\colon\{0,1\}^n\to\{0,1\}$ that approximates f well under the uniform distribution in the sense that $f(x)\neq g(x)$ for at most $\varepsilon\cdot 2^n$ inputs x. Suppose also that $\boldsymbol{X}$ is a random variable that ε-fools g. Do we have that $\boldsymbol{X}$ ε-fools f? The answer is no, unfortunately—we cannot conclude that $\boldsymbol{X}$ fools f, even mildly.



Here the shaded region represents the set of inputs on which f and g disagree, all $\leqslant\varepsilon\cdot 2^n$ of them. Now $\boldsymbol{X}$ is supported on $2^{\text{seed length}}$ points. We are usually interested in cases where the seed length is small, but now suppose it's large; suppose it is supported on $n/10$ points. Then $\boldsymbol{X}$ is supported on $2^{n/10}$ points, and it is possible that all of these points lie in the shaded error region—in particular this happens if $2^{n/10}<\varepsilon\cdot 2^n$. In this case, $\boldsymbol{X}$ does not ε-fool f, and so fooling an ordinary approximator doesn't necessarily fool f.

This news is somewhat sad, but fortunately there is good news: if f is approximable in a stronger (sandwiching) sense, then fooling the sandwiching approximator does fool f.

Definition 8.10 (δ-sandwich)**.** *Let $f,f_\ell,f_u\colon\{0,1\}^n\to\mathbb{R}$. Then f is* δ-sandwiched *by (f_ℓ,f_u) if:*

1. *for all $x\in\{0,1\}^n$, $f_\ell(x)\leqslant f(x)\leqslant f_u(x)$, and*

[6] That is, for every parity of size at most k, it fools it to within ε.

2. the expectation $\mathbb{E}_{\boldsymbol{x}\sim\boldsymbol{X}}[f_u(\boldsymbol{x}) - f_\ell(\boldsymbol{x})] \leqslant \delta$.

The first condition is the sandwiching condition, and the second condition is the error condition.

With this notion of approximation, then fooling f_ℓ and f_u does fool f.

Lemma 8.11 (Sandwiching lemma)**.** *Suppose that* $f\colon \{0,1\}^n \to \mathbb{R}$ *is* δ*-sandwiched by* (f_ℓ, f_u)*, and suppose that a random variable* $\boldsymbol{X}$ *over* $\{0,1\}^n$ *both* ε*-fools* f_ℓ *and* ε*-fools* f_u*. Then* $\boldsymbol{X}$ $(\varepsilon+\delta)$*-fools* f*.*

Proof. We have

$$\begin{aligned}\mathbb{E}_{\boldsymbol{x}\sim\boldsymbol{X}}[f(\boldsymbol{x})] &\leqslant \mathbb{E}_{\boldsymbol{x}\sim\boldsymbol{X}}[f_u(\boldsymbol{x})] \\ &\leqslant \mathbb{E}_{\boldsymbol{u}\sim\mathcal{U}}[f_u(\boldsymbol{u})] + \varepsilon \\ &\leqslant \mathbb{E}_{\boldsymbol{u}\sim\mathcal{U}}[f(\boldsymbol{u})] + \varepsilon + \delta.\end{aligned}$$

The proof for the lower bound is similar, as we can readily show that

$$\mathbb{E}_{\boldsymbol{x}\in\boldsymbol{X}}[f(\boldsymbol{x})] \geqslant \cdots \geqslant \mathbb{E}_{\boldsymbol{u}\sim\mathcal{U}}[f(\boldsymbol{u})] - \varepsilon - \delta. \qquad \square$$

Let us now think about this in the context of k-wise independent distributions: sandwiching polynomials are useful for pseudorandomness.

Corollary 8.12 (Sandwiching polynomials give PRGs)**.** *A function* $f\colon \{0,1\}^n \to \mathbb{R}$ *is* ε*-fooled by any* k*-wise independent distribution* $\boldsymbol{X}$ *if there exist* ε*-sandwiching real polynomials* q_ℓ, q_u *of degree* k *such that* $q_\ell \leqslant f \leqslant q_u$ *and* $\mathbb{E}_{x\sim\boldsymbol{X}}[q_u(x) - q_\ell(x)] \leqslant \varepsilon$*.*

Proof. Exercise. $\square$

§8.4 Fooling degree-d $\mathbb{F}_2$-polynomials via ε-biased generators: Viola's theorem

Recall the class we introduced in the very first lecture:

$$\begin{aligned}\mathrm{DEG}_d &= \mathrm{DEG}_{d(n)} \\ &= \{\text{all } f : \{0,1\}^n \to \{0,1\} \text{ computable by a degree-}d\ \mathbb{F}_2\text{-polynomial}\}.\end{aligned}$$

Also recall that we have already discussed how we might fool degree-d real polynomials using d-wise independence with seed length $O(d \log n)$ to perfect accuracy (i.e. $\varepsilon = 0$). Howeverm ooling DEG_d is an entirely different matter! Let's highlight this via a simple example.

Example 8.13. Let the random variable $\boldsymbol{X} = (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n)$, where $\boldsymbol{X}$ is generated by tossing $n-1$ fair coins and letting the last coin be the parity of the first $n-1$ coins:

$$\begin{aligned}\boldsymbol{X}_1 &= \mathcal{U}_1 \\ \boldsymbol{X}_2 &= \mathcal{U}_2 \\ &\vdots\end{aligned}$$

$$\boldsymbol{X}_{n-1} = \mathcal{U}_{n-1}$$
$$\boldsymbol{X}_n = \mathcal{U}_1 \oplus \mathcal{U}_2 \oplus \cdots \oplus \mathcal{U}_{n-1}.$$

Then the seed length of $\boldsymbol{X}$ is $n-1$, and $\boldsymbol{X}$ is $(n-1)$-wise independent. This is very large, but this random variable completely fails to fool even $\mathrm{PARITY}_{[n]} \in \mathrm{DEG}_1$, because for every outcome of $\boldsymbol{X}$, the parity of the n bits of $\boldsymbol{X}$ is always 0. So $\boldsymbol{X}$ does not ε-fool $\mathrm{PARITY}_{[n]}$ for any ε.
Therefore $(n-1)$-wise independence completely fails to fool even degree-1 polynomials.

Some motivation Why should we even try to fool degree-d polynomials even if we only care about Boolean circuits? One motivation is that we provably need to fool degree-d polynomials in order to improve the state of the art for our current PRGs for Boolean circuits, as the following theorem shows.

Claim 8.14. *An ε-PRG against DEG_d for $d = (\log n)^{\omega(1)}$ immediately gives us a 3ε-PRG against $\mathrm{AC}^0(\oplus)$ circuits[7] of depth d.*

We will prove this claim, with the help of the following fact: for any circuit of the kind we're interested in, there is a distribution over polynomials of low degree such that for any fixed input, a random polynomial from this distribution agrees with the circuit on that input with sufficiently high probability.

Fact 8.15. *For any $O(1)$-depth, $\mathrm{poly}(n)$-size circuit C, there exists a distribution $\mathcal{P}$ over degree-$(\mathrm{polylog}(n))$ $\mathbb{F}_2$-polynomials such that for all $z \in \mathbb{F}_2^n$,*

$$\Pr_{\boldsymbol{p}\sim\mathcal{P}}[\boldsymbol{p}(z) = C(z)] \geqslant 1 - \varepsilon,$$

where $\varepsilon = 2^{-\Omega(n)}$.

Proof. Unrequired exercise. □

Proof of Claim 8.14. Let $G\colon \{0,1\}^s \to \{0,1\}^n$ be an ε-PRG against DEG_d, so that by the fact above,

$$\Pr_{\boldsymbol{u}\sim\mathcal{U}_s}[C(G(\boldsymbol{u})) = 1] \approx_\varepsilon \Pr_{\substack{\boldsymbol{u}\sim\mathcal{U}_s \\ \boldsymbol{p}\sim\mathcal{P}}}[\boldsymbol{p}(G(\boldsymbol{u})) = 1].$$

Now since G is an ε-PRG for DEG_d and d is greater than the degree of the polynomials in $\mathcal{P}$, we have that

$$\Pr_{\substack{\boldsymbol{u}\sim\mathcal{U}_s \\ \boldsymbol{p}\sim\mathcal{P}}}[\boldsymbol{p}(G(\boldsymbol{u})) = 1] \approx_\varepsilon \Pr_{\substack{\boldsymbol{u}\sim\mathcal{U}_n \\ \boldsymbol{p}\sim\mathcal{P}}}[\boldsymbol{p}(\boldsymbol{u}) = 1].$$

Since $\mathcal{P}$ fools C input-by-input, we have

$$\Pr_{\substack{\boldsymbol{u}\sim\mathcal{U}_n \\ \boldsymbol{p}\sim\mathcal{P}}}[\boldsymbol{p}(\boldsymbol{u}) = 1] \approx_\varepsilon \Pr_{\boldsymbol{u}\sim\mathcal{U}_n}[C(\boldsymbol{u}) = 1].$$

[7] $\mathrm{AC}^0(\oplus)$ is the class of all $\mathrm{poly}(n)$-size, $O(1)$-depth circuits with AND, OR, $\neg$, and $\oplus$ gates.

Combining these three facts, we have that

$$\Pr_{\boldsymbol{u}\sim\mathcal{U}_s}[C(G(\boldsymbol{u}))=1] \approx_{3\varepsilon} \Pr_{\boldsymbol{u}\sim\mathcal{U}_n}[C(\boldsymbol{u})=1],$$

which is the same as saying that G is a 3ε-PRG for $\mathrm{AC}^0(\oplus)$ circuits of depth d. □

So we actually have a grounded reason—beyond sheer love and longing, of course—to obtain PRGs for DEG_d for nice d.

We have already seen that there is a bit of a stumbling block because we don't know how to get good-enough correlation bounds for $\mathbb{F}_2$-polynomials particularly for $d > \log n$. But we did have some correlation bounds for degree up to $\log n$, and it turns out that the ideas in those correlation bounds can help us prove good PRGs for degree up to $\log n$. There is a line of work for this kind of thing that culminated in the work of Emanuele Viola, who proved that the sum of d-independent δ-biased random variables fools degree-d $\mathbb{F}_2$-polynomials:

Theorem 8.1 (Viola's theorem, [Vio09])**.** *Let $\boldsymbol{Y}_1,\dots,\boldsymbol{Y}_d$ be independent δ-biased random variables over $\{0,1\}^n$ for small $\delta \leqslant \frac{1}{2}$. Define $\boldsymbol{Y} \coloneqq \boldsymbol{Y}_1+\ldots+\boldsymbol{Y}_d$. Then $\boldsymbol{Y}$ indeed $4(\frac{\delta}{2})^{1/2^{d-1}}$-fools DEG_d.*

In particular, with $\varepsilon = 4(\frac{\delta}{2})^{1/2^{d-1}}$, we get a PRG with seed length

$$O\left(d\cdot\log\left(\frac{n}{\delta}\right)\right) = O\left(d\cdot\log n + d\cdot 2^d\cdot\log\frac{1}{\varepsilon}\right).$$

Immediately we see that this seed length breaks down to the trivial case when $d=\log n$, which tracks with our discussion above.

Our next question might be, why is summing independent δ-biased random variables a good idea? The answer is in the following fact.

Fact 8.16. *Let $\boldsymbol{Y}_1,\dots,\boldsymbol{Y}_d$ be independent δ-biased random variables over $\mathbb{F}_2^n$. Then for any d, $\boldsymbol{Y}=\boldsymbol{Y}_1+\ldots+\boldsymbol{Y}_d$ is δ^d-biased.*

Proof. Suppose $S\subseteq[n]$ with $S\neq\emptyset$. Then

$$\begin{aligned}
\left|\mathbb{E}_{\boldsymbol{Y}}\left[\chi_S\left(\sum_{i=1}^d \boldsymbol{Y}_i\right)\right]\right| &= \left|\mathbb{E}\left[(-1)^{\sum_{j\in S}\sum_{i=1}^d \boldsymbol{Y}_{i,j}}\right]\right| && \text{by the definition of } \chi_S\\
&= \left|\mathbb{E}\left[(-1)^{\sum_{i=1}^d\sum_{j\in S} \boldsymbol{Y}_{i,j}}\right]\right|\\
&= \left|\prod_{i=1}^d \mathbb{E}_{\boldsymbol{Y}}\left[\chi_S(\boldsymbol{Y}_i)\right]\right| && \text{by independence}\\
&\leqslant \delta^d,
\end{aligned}$$

where the last inequality follows from the fact that each $\boldsymbol{Y}_i$ is δ-biased. □

So the sum of the $\boldsymbol{Y}_i$ improves the bias, and so there is hope that that this sum may be be useful for even more things, in particular, helping us to fool degree-d polynomials towards the result of Viola's.

We will now prove Viola's theorem. The high-level strategy is to apply a sensible inductive hypothesis: we will show that if we can fool DEG_{d-1} somehow, then we can add one more δ-biased random variable to fool DEG_d (with somewhat worse parameters). The key lemma which instantiates this strategy is as follows.

Lemma 8.17 (Key lemma for Theorem 8.1)**.** *Suppose that $\boldsymbol{W}$ fools DEG_{d-1} with error γ, and suppose that $\boldsymbol{Y}$ is a δ-biased random variable over $\{0,1\}^n$ that is independent of $\boldsymbol{W}$. Then $\boldsymbol{W}+\boldsymbol{Y}$ fools DEG_d with error $\sqrt{2\gamma}+\frac{\delta}{2}$.*

Assuming this lemma holds, we can prove Viola's theorem as follows.

Proof of Theorem 8.1. By the definition of δ-biased random variables, Y_1 $\frac{\delta}{2}$-fools DEG_1. Let $\varepsilon_1 = \frac{\delta}{2}$. Furthermore, let $\varepsilon_{i+1} = \sqrt{2\varepsilon_i} + \frac{\delta}{2}$ for $i \geqslant 1$. Then by Lemma 8.17, $\boldsymbol{Y}_1 + \boldsymbol{Y}_2$ ε_2-fools DEG_2, and by induction, $\boldsymbol{Y}$ ε_d-fools DEG_d. Now $\delta \leqslant \frac{1}{2}$, so

$$\varepsilon_{i+1} \leqslant \sqrt{2\varepsilon_i} + \frac{1}{2}\sqrt{\frac{\delta}{2}} \underset{\delta/2 \leqslant \varepsilon_i}{\leqslant} \left(\sqrt{2}+\frac{1}{2}\right)\sqrt{\varepsilon_i} \leqslant 2\sqrt{\varepsilon_i}.$$

Therefore,

$$\begin{aligned}
\varepsilon_2 &\leqslant 2 \cdot \left(\frac{\delta}{2}\right)^{1/2} \\
\varepsilon_3 &\leqslant 2^{1+\frac{1}{2}} \cdot \left(\frac{\delta}{2}\right)^{1/2^2} \\
\varepsilon_4 &\leqslant 2^{1+\frac{1}{2}+\frac{1}{2^2}} \cdot \left(\frac{\delta}{2}\right)^{1/2^3} \\
&\vdots \\
\varepsilon_d &\leqslant 2^{1+\frac{1}{2}+\frac{1}{2^2}+\ldots+\frac{1}{2^{d-1}}} \cdot \left(\frac{\delta}{2}\right)^{1/2^{d-1}},
\end{aligned}$$

and since $\sum_{i=0}^{d-1} \frac{1}{2^i} = 2 - \frac{1}{2^{d-1}}$, we have that

$$\varepsilon_d \leqslant 4\left(\frac{\delta}{2}\right)^{1/2^{d-1}},$$

as desired. □

So we only need to prove Lemma 8.17. The proof of this lemma involves a case analysis depending on the "imbalance" of the degree-i function $f \in \mathrm{DEG}_i$ that is being fooled.

Definition 8.18 (Imbalance of a Boolean function)**.** *For a function $f\colon \{0,1\}^n \to \{0,1\}$, the* imbalance *of f is*

$$\mathsf{imbal}(f) \coloneqq \left|\mathbb{E}_{\boldsymbol{u}\sim\mathcal{U}_n}\left[(-1)^{f(\boldsymbol{u})}\right]\right| = 2\left|\mathbb{E}[f] - \frac{1}{2}\right| \in [0,1].$$

Intuitively, we can see that:

- If f has large $\mathsf{imbal}(f)$, then $\boldsymbol{W}$ already fools f well, and so adding $\boldsymbol{Y}$ doesn't change much.
- If f has small $\mathsf{imbal}(f)$, then we will do an analysis that is similar to the one we did for the correlation bounds for $\mathbb{F}_2$-polynomials of degree up to $\log n$; we'll do squaring of the correlation, take derivatives, use Cauchy-Schwarz, and so on.

The following two lemmas handle both the (almost) balanced and imbalanced cases.

Lemma 8.19 (Imbalanced case)**.** *Suppose a random variable $\boldsymbol{W}$ γ-fools DEG_{i-1}. Then $\boldsymbol{W}$ also fools any $f \in \mathrm{DEG}_i$ with error $\gamma/\mathsf{imbal}(f)$.*

Lemma 8.20 (Balanced case)**.** *Suppose a random variable $\boldsymbol{W}$ γ-fools DEG_{i-1}, and let the δ-biased random variable $\boldsymbol{Y}$ be independent of $\boldsymbol{W}$. Then $\boldsymbol{W} + \boldsymbol{Y}$ fools any $f \in \mathrm{DEG}_i$ with error*

$$\mathsf{imbal}(f) + \sqrt{\frac{\gamma}{2}} + \frac{\delta}{2}.$$

These two lemmas essentially say that if we're trying to fool a pretty imbalanced arbitrary function, then we can do so using only $\boldsymbol{W}$; if we're trying to fool an almost balanced function, then we can do so by adding $\boldsymbol{Y}$ to $\boldsymbol{W}$ with some loss tradeoff. These two lemmas are the key to proving Lemma 8.17, as we now see.

Write $f^{+y}(x)$ to mean $f(x+y)$.

Proof of Lemma 8.17. Fix any $f \in \mathrm{DEG}_i$. Fix any outcome of $\boldsymbol{Y}$. Then the function $f^{+Y}(x) \coloneqq f(x+Y)$ is a degree-i $\mathbb{F}_2$-polynomial in $x_1, \ldots, x_n$, and $\mathsf{imbal}(f^{+Y}) = \mathsf{imbal}(f)$, since this function is merely a translation of f.

Therefore, by Lemma 8.19, $\boldsymbol{W} + \boldsymbol{Y}$ fools f with error $\gamma/\mathsf{imbal}(f)$. And by Lemma 8.20, $\boldsymbol{W} + \boldsymbol{Y}$ fools f with error $\mathsf{imbal}(f) + \sqrt{\frac{\gamma}{2}} + \frac{\delta}{2}$. So it must be that $\boldsymbol{W} + \boldsymbol{Y}$ fools f with error

$$\min\left\{\frac{\gamma}{\mathsf{imbal}(f)}, \mathsf{imbal}(f) + \sqrt{\frac{\gamma}{2}} + \frac{\delta}{2}\right\} \leqslant \sqrt{2\gamma} + \frac{\delta}{2},$$

as desired. □

So, stepping back, we only need to show Lemmas 8.19 and 8.20, and both of their proofs will involve taking derivatives.

Recall that for $y \in \mathbb{F}_2^n$ and for $f \colon \mathbb{F}_2^n \to \mathbb{F}_2$, we define the directional derivative of f at y, $\partial_y f \colon \mathbb{F}_2^n \to \mathbb{F}_2$, by

$$\begin{aligned}\partial_y f(x) &= f^{+y}(x) + f(x) \\ &= f(x+y) + f(x).\end{aligned}$$

We have already seen that if f is of degree i, then for all $y \in \mathbb{F}_2^n$, $\partial_y f$ is of degree at most $i-1$. So now we can prove Lemma 8.19.

Proof of Lemma 8.19. We want to show that, for uniform $\mathcal{U}'$,

$$\mathsf{imbal}(f) \cdot \left| \mathbb{E}_{\boldsymbol{w} \sim \boldsymbol{W}} \left[(-1)^{f(\boldsymbol{w})} \right] - \mathbb{E}_{\boldsymbol{u}' \sim \mathcal{U}'} \left[(-1)^{f(\boldsymbol{u}')} \right] \right| \leqslant 2\gamma.$$

Let $\mathcal{U}$ be independent (of every other source of randomness) and uniform. Then since, by definition, $\mathsf{imbal}(f) = \mathbb{E}[(-1)^{f(\mathcal{U})}]$, we have that

$$\begin{aligned}
\mathsf{imbal}(f) \left| \mathbb{E}\left[(-1)^{f(\boldsymbol{w})} \right] - \mathbb{E}\left[(-1)^{f(\boldsymbol{u}')} \right] \right| &= \left| \mathbb{E}\left[(-1)^{f(\boldsymbol{w}) + f(\mathcal{U})} \right] - \mathbb{E}\left[(-1)^{f(\boldsymbol{u}') + f(\mathcal{U})} \right] \right| \\
&= \left| \mathbb{E}\left[(-1)^{f(\boldsymbol{w}) + f(\boldsymbol{w} + \mathcal{U})} \right] - \mathbb{E}\left[(-1)^{f(\boldsymbol{u}') + f(\boldsymbol{u}' + \mathcal{U})} \right] \right| \\
&= \left| \mathbb{E}\left[(-1)^{\partial_{\mathcal{U}} f(\boldsymbol{w})} \right] - \mathbb{E}\left[(-1)^{\partial_{\mathcal{U}} f(\boldsymbol{u}')} \right] \right| \\
&\leqslant 2\gamma,
\end{aligned}$$

since for any fixing of $\mathcal{U}$, $\partial_{\mathcal{U}} f$ is a degree-$(i-1)$ function, and hence is γ-foolable by $\boldsymbol{W}$. □

Next time, we will see the proof of Lemma 8.20, and move on to more. Enjoy Spring Break!

§9 Lecture 09—19th March, 2024

Last time. In the previous class, we:

- Saw some basic Fourier analysis over the Boolean hypercube, as well as some applications in fooling size-s decision trees, and fooling k-juntas better.
- Discussed why it is smart to fool sandwiching approximators.
- Introduced Viola's theorem: the sum of d-many ε-biased random variables fools degree-d polynomials.

Today. We will:

- Finish the proof of Viola's theorem.
- Introduce Braverman's theorem [Bra08]: $(\log n)^{O(1)}$-wise independence fools AC^0 circuits. Some tools towards this:
  - Beigel-Reingold-Spielman's theorem [BRS91]: AC^0 circuits can be approximated by low-degree polynomials.
  - Linial-Mansour-Nisan's theorem [LMN93]: AC^0 circuits can be approximated by low-degree polynomials with small Fourier mass.
  - Hopefully start the proof of Braverman's theorem.

Reminders: the second problem set is due tomorrow, and the project progress report is due Thursday April 4th.

---

§9.1 Finishing the proof of Viola's theorem

Recall our statement of Viola's theorem from before spring break:

Theorem 9.1 (Viola's theorem, [Vio09])**.** *Let $\boldsymbol{Y}_1, \dots, \boldsymbol{Y}_d$ be independent δ-biased random variables over $\{0,1\}^n$ for small $\delta \leqslant \frac{1}{2}$. Define $\boldsymbol{Y} \coloneqq \boldsymbol{Y}_1 + \dots + \boldsymbol{Y}_d$. Then $\boldsymbol{Y}$ indeed $4(\frac{\delta}{2})^{1/2^{d-1}}$-fools DEG_d.*

In particular, with $\varepsilon = 4(\frac{\delta}{2})^{1/2^{d-1}}$, we get a PRG with seed length

$$O\left(d \cdot \log\left(\frac{n}{\delta}\right)\right) = O\left(d \cdot \log n + d \cdot 2^d \cdot \log \frac{1}{\varepsilon}\right).$$

We were left needing to prove the following lemma:

Lemma 9.1 (Balanced case)**.** *Suppose a random variable $\boldsymbol{W}$ γ-fools DEG_{i-1}, and let the δ-biased random variable $\boldsymbol{Y}$ be independent of $\boldsymbol{W}$. Then $\boldsymbol{W} + \boldsymbol{Y}$ fools any $f \in \mathrm{DEG}_i$ with error*

$$\mathsf{imbal}(f) + \sqrt{\frac{\gamma}{2}} + \frac{\delta}{2}.$$

Towards a proof of this lemma, we present the following claim which is useful in the context of the proof of Lemma 9.1. It says that for a function g, multiplying two copies of g on separate random variables allows us to relax the amount of randomness necessary on half of the inputs.

Claim 9.2. *Fix an even $n \in \mathbb{N}$. Let $g\colon \mathbb{F}_2^{n/2} \to \{\pm 1\}$ and $f(x,y) = g(x) \cdot g(y) \in \{\pm 1\}$ for $x, y \in \mathbb{F}_2^{n/2}$, where $\cdot$ denotes XOR. Then for $\boldsymbol{u} \sim \mathcal{U}_{n/2}$ with $\boldsymbol{Y}$ any ε-biased random variable over $\mathbb{F}_2^{n/2}$, we have that the distribution $(\boldsymbol{u}, \boldsymbol{u} + \boldsymbol{Y})$ fools f with error ε, i.e.*

$$\left|\mathbb{E}_{\boldsymbol{u},\boldsymbol{u}' \sim \mathcal{U}_{n/2}}[f(\boldsymbol{u}, \boldsymbol{u}')] - \mathbb{E}_{\boldsymbol{u},\boldsymbol{u}' \sim \mathcal{U}_{n/2}}[f(\boldsymbol{u}, \boldsymbol{u}' + \boldsymbol{Y})]\right| \leqslant \varepsilon.$$

Proof. Consider the function $F\colon \mathbb{F}_2^{n/2} \to [-1,1]$ defined as

$$F(x) \coloneqq \mathbb{E}_{\boldsymbol{u} \sim \mathcal{U}_{n/2}}[g(\boldsymbol{u}) \cdot g(\boldsymbol{u} + x)] = \mathbb{E}_{\boldsymbol{u} \sim \mathcal{U}_{n/2}}[f(\boldsymbol{u}, \boldsymbol{u} + x)].$$

To prove the claim, it suffices to show that $\boldsymbol{Y}$ ε-fools F. We will show that the Fourier L_1-norm of F is $L_1(F) = 1$; then as we saw from Lemma 8.6, an ε-biased random variable $\boldsymbol{Y}$ fools F with error $\varepsilon \cdot L_1(F) = \varepsilon$.

Towards this, fix any $S \subseteq [n/2]$. Recalling that $\widehat{F}(S) = \mathbb{E}_{\boldsymbol{u} \sim \mathcal{U}_{n/2}}[F(\boldsymbol{u}) \cdot \chi_S(\boldsymbol{u})]$, we have

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{u},\boldsymbol{u}' \sim \mathcal{U}_{n/2}}[f(\boldsymbol{u}, \boldsymbol{u}' + \boldsymbol{u}) \cdot \chi_S(\boldsymbol{u}.)] &= \mathbb{E}_{\boldsymbol{u},\boldsymbol{u}' \sim \mathcal{U}_{n/2}}[g(\boldsymbol{u}) \cdot g(\boldsymbol{u}' + \boldsymbol{u}) \cdot \chi_S(\boldsymbol{u}')] \\
&= \mathbb{E}_{\boldsymbol{u},\boldsymbol{u}' \sim \mathcal{U}_{n/2}}[g(\boldsymbol{u}) \cdot g(\boldsymbol{u}') \cdot \chi_S(\boldsymbol{u} + \boldsymbol{u}')] \\
&= \mathbb{E}_{\boldsymbol{u},\boldsymbol{u}' \sim \mathcal{U}_{n/2}}[g(\boldsymbol{u}) \cdot g(\boldsymbol{u}') \cdot \chi_S(\boldsymbol{u}) \cdot \chi_S(\boldsymbol{u}')] \\
&= \left(\mathbb{E}_{\boldsymbol{u} \sim \mathcal{U}_{n/2}}[g(\boldsymbol{u}) \cdot \chi_S(\boldsymbol{u})]\right)^2 = \widehat{g}(S)^2.
\end{aligned}$$

Therefore

$$L_1(F) = \sum_{S \subseteq [n/2]} |\widehat{F}(S)| = \sum_{S \subseteq [n/2]} |\widehat{g}(S)|^2 = \mathbb{E}_{\boldsymbol{u} \sim \mathcal{U}_{n/2}}[g(\boldsymbol{u})^2] = 1,$$

since $g(\boldsymbol{u}) \in \{\pm 1\}$. □

Now we can prove Lemma 9.1.

Proof of Lemma 9.1. We're interested in how $\boldsymbol{W}+\boldsymbol{Y}$ fools $f \in \mathrm{DEG}_i$, and so we start with

$$\begin{aligned}|\mathbb{E}[f(\boldsymbol{W}+\boldsymbol{Y})] - \mathbb{E}[f(\mathcal{U})]| &= \frac{1}{2}\left|\mathbb{E}\left[(-1)^{f(\boldsymbol{W}+\boldsymbol{Y})}\right] - \underbrace{\mathbb{E}\left[(-1)^{f(\mathcal{U})}\right]}_{\mathsf{imbal}(f)}\right| \\ &\leqslant \frac{1}{2}\left|\mathbb{E}\left[(-1)^{f(\boldsymbol{W}+\boldsymbol{Y})}\right]\right| + \frac{1}{2}\cdot\mathsf{imbal}(f),\end{aligned}$$

where the inequality follows from the triangle inequality. We will now bound the first term in the sum above, by squaring and using the Cauchy-Schwarz inequality:

$$\begin{aligned}\left(\mathbb{E}\left[(-1)^{f(\boldsymbol{W}+\boldsymbol{Y})}\right]\right)^2 &\leqslant \mathbb{E}_{\boldsymbol{w}\sim\boldsymbol{W}}\left[\mathbb{E}_{\boldsymbol{y}\sim\boldsymbol{Y}}\left[(-1)^{f(\boldsymbol{w}+\boldsymbol{y})}\right]\right]^2 \\ &= \mathbb{E}_{\boldsymbol{w}\sim\boldsymbol{W},\boldsymbol{y},\boldsymbol{y}'\sim\boldsymbol{Y}}\left[(-1)^{f(\boldsymbol{w}+\boldsymbol{y})}\cdot(-1)^{f(\boldsymbol{w}+\boldsymbol{y}')}\right] \\ &= \mathbb{E}_{\boldsymbol{w}\sim\boldsymbol{W},\boldsymbol{y},\boldsymbol{y}'\sim\boldsymbol{Y}}\left[(-1)^{f(\boldsymbol{w}+\boldsymbol{y})+f(\boldsymbol{w}+\boldsymbol{y}')}\right],\end{aligned}$$

since $\boldsymbol{y}'$ is independent but identically distributed as $\boldsymbol{y}$. We can now apply the claim to the function $f(\boldsymbol{w}+\cdot)$. Now, for any fixed $\boldsymbol{Y}$ outcome, $f^{+\boldsymbol{Y}}(x) = f(x+\boldsymbol{Y})$ is a degree-i polynomial in x, and for any fixed outcome of $\boldsymbol{Y}+\boldsymbol{Y}'$, by our arguments in previous lectures, $f(x+\boldsymbol{Y})+f(x+\boldsymbol{Y}')$ is the directional derivative $\partial_{\boldsymbol{Y}+\boldsymbol{Y}'}f^{+\boldsymbol{Y}}(x)$ of f in the direction $\boldsymbol{Y}+\boldsymbol{Y}'$. Therefore, we must have $\deg\left(f(x+\boldsymbol{Y})+f(x+\boldsymbol{Y}')\right) \leqslant i-1$. Since $\boldsymbol{W}$ γ-fools DEG_{i-1}, we have

$$\mathbb{E}_{\boldsymbol{w}\sim\boldsymbol{W},\boldsymbol{y},\boldsymbol{y}'\sim\boldsymbol{Y}}\left[(-1)^{f(\boldsymbol{w}+\boldsymbol{y})+f(\boldsymbol{w}+\boldsymbol{y}')}\right] \leqslant \mathbb{E}_{\boldsymbol{u}\sim\mathcal{U}_n,\boldsymbol{y},\boldsymbol{y}'\sim\boldsymbol{Y}}\left[(-1)^{f(\boldsymbol{u}+\boldsymbol{y})+f(\boldsymbol{u}+\boldsymbol{y}')}\right] + 2\gamma.$$

Now note that the distribution of $(\boldsymbol{u}+\boldsymbol{y},\boldsymbol{u}+\boldsymbol{y}')$ is the same as the distribution of $(\boldsymbol{u},\boldsymbol{u}+\boldsymbol{y}+\boldsymbol{y}')$, since $\boldsymbol{u}$ is uniform and the second is the shift of the first coordinate by $\boldsymbol{y}+\boldsymbol{y}'$, which is δ^2-biased. Therefore, by Claim 9.2,

$$\mathbb{E}_{\boldsymbol{u}\sim\mathcal{U}_n,\boldsymbol{y},\boldsymbol{y}'\sim\boldsymbol{Y}}\left[(-1)^{f(\boldsymbol{u}+\boldsymbol{y})+f(\boldsymbol{u}+\boldsymbol{y}')}\right] \leqslant \mathbb{E}_{\boldsymbol{u}\sim\mathcal{U}_n}\left[(-1)^{f(\boldsymbol{u})}\right]^2 + \delta^2 = \mathsf{imbal}(f)^2 + \delta^2.$$

Putting the pieces together, we have

$$\begin{aligned}|\mathbb{E}[f(\boldsymbol{W}+\boldsymbol{Y})] - \mathbb{E}[f(\mathcal{U})]| &\leqslant \frac{1}{2}\left|\mathbb{E}\left[(-1)^{f(\boldsymbol{W}+\boldsymbol{Y})}\right]\right| + \frac{1}{2}\cdot\mathsf{imbal}(f) \\ &\leqslant \frac{1}{2}\sqrt{\mathsf{imbal}(f)^2+\delta^2+2\gamma} + \frac{1}{2}\cdot\mathsf{imbal}(f) \\ &\leqslant \frac{1}{2}\cdot\left(\mathsf{imbal}(f)+\delta+\sqrt{2\gamma}\right) + \frac{1}{2}\cdot\mathsf{imbal}(f) \\ &= \mathsf{imbal}(f) + \frac{\delta}{2} + \sqrt{\frac{\gamma}{2}},\end{aligned}$$

where the last inequality is because $\sqrt{a+b+c} \leqslant \sqrt{a}+\sqrt{b}+\sqrt{c}$ for $a,b,c \geqslant 0$. This completes the proof of the lemma. □

§9.2 Fooling AC^0 circuits with k-wise independent RVs: Braverman's theorem

We now move on to discussing Braverman's theorem [Bra08], which states that about $(\log n)^{O(1)}$-wise independence fools AC^0 circuits. Although we will not go through the entire proof today, we will discuss some setup and build up towards it.

First, some motivation. We love AC^0 circuits because they are the simplest non-trivial circuits, and they are computationally powerful, and so we are interested in understanding how to fool them, get lower bounds on them, etc. We already got worst-case and average-case lower bounds against AC^0, and so we are ready to take the relationship to the next level: we want to understand how to fool AC^0 circuits, construct PRGs against AC^0, etc. There are many ways to fool AC^0 circuits. We will see one of them that is not the simplest, isn't the best, and doesn't give the best parameters, but uses the most generic machinery: k-wise independence.

There's some interesting history that motivates why someone might want to do this. In 1990, Linial and Nisan made the following conjecture:

Conjecture 9.3 (Linial-Nisan)**.** *Every* $\mathrm{poly}(n)$*-size,* $O(1)$*-depth circuit is fooled by any* k*-wise independent distribution for* $k = \mathrm{polylog}(n)$*.*

No one made any progress on this until 2007, when Louis Bazzi proved the case where $d = 2$. The proof was quite complicated, but only one year later, Alexander Razborov dramatically simplified Bazzi's result, and just around the same time, Mark Braverman [Bra08] proved the case for general d.

We will prove the following version of Braverman's theorem. It is not quite the best possible; the best possible is achieved via the multi-switching lemma rather than the switching lemma due to Håstad.

Theorem 9.2 (Braverman's theorem, [Bra08])**.** *Let* $k = \left(\log \frac{s}{\varepsilon}\right)^{O(d^2)}$*, and let* $\boldsymbol{X} \sim \mathcal{D}$*, where* $\mathcal{D}$ *is a* k*-wise independent distribution over* $\{0,1\}^n$*. Then for any* AC^0 *circuit* C *of size* s *and depth* d*,* $\boldsymbol{X}$ *fools* C *with error* ε*.*

Again, this is not the state of the art; the state of the art has $k = (\log s)^{O(d)} \cdot \log \frac{1}{\varepsilon}$.

Before we prove this, why might we believe that Conjecture 9.3 is true, that this k-wise independence is enough to fool AC^0 circuits? We saw that:

- k-wise independence fools k-juntas, depth-d decision trees, etc...
- k-wise independence does also fool real polynomials of degree k, almost by definition.

It was known since the 1980s that AC^0 circuits are well-approximated by low-degree real polynomials in two different senses, but each of the two notions of approximation for AC^0 circuits by low-degree polynomials is not quite enough to give us PRGs, and Braverman exhibited a clever combination of them to get sandwiching polynomial approximators which we already saw are good enough for fooling.

For the rest of today, we will discuss the two notions of approximation of AC^0 circuits by low-degree polynomials, and maybe then we can start the proof of Braverman's theorem.

§9.2.1 Polynomially-approximating AC^0: Beigel-Reingold-Spielman's theorem

The first notion of approximation of AC^0 circuits by low-degree polynomials, a "pointwise" notion of approximation, is due to the ideas of Beigel, Reingold, and Spielman [BRS91].

Lemma 9.4 (BRS theorem, [BRS91])**.** *Let $f \in \mathrm{AC}^0_{s,d}$. Let $\mathcal{D}$ be any distribution over $\{0,1\}^n$. Then there exists a real polynomial p such that*

$$\Pr_{\boldsymbol{x}\sim\mathcal{D}}[p(x) = f(x)] \geqslant 1 - \varepsilon,$$

where

(i) $\deg(p) \leqslant \left(\log \frac{s}{\varepsilon}\right)^{O(d)}$*, and*

(ii) for every input $x \in \{0,1\}^n$, $|p(x)| \leqslant \exp\left(\left(\log \frac{s}{\varepsilon}\right)^{O(d)}\right)$.

Before we prove this, observe that this alone is not good enough for us; this p may be bad on average, since we only have 0 error a $1-\varepsilon$ fraction of the time, but the remaining fraction of the time the average error of p on f could be large.

Proof of Lemma 9.4. We will prove this lemma in two steps. First, we will prove it for the case where $d = 1$, i.e. for depth-1 circuits:

$$f = x_1 \vee x_2 \vee \ldots \vee x_t = \bigvee_{i=1}^{t} x_t.$$

Consider such an f as above. We will describe a distribution over polynomials such that for every fixed input, a random draw from that distribution has a high probability of being equal to f on that input.

Let $V_0 = \{x_1, \ldots, x_t\}$ be the set of variables, and define some initial polynomial $p_0(x) = x_1 + \ldots + x_t$. Then for $i = 1, \ldots, \log_2 t + 1$, let:

- V_i be constructed from V_{i-1} by discarding each variable with probability $\frac{1}{2}$.
- $p_i = \sum_{x_j \in V_i} x_j$.

So $p_0, p_1, \ldots, p_{\log_2 t+1}$ are all polynomials of degree 1 taking values in $[0, t]$ on any input in $\{0,1\}^t$. Now fix any input assignment $z \in \{0,1\}^t$ such that some $z_i = 1$, that is, $z \neq 0^t$, so $z_1 \vee \ldots \vee z_t = 1$.

Claim 9.5. *Let E be the event that at least one of $p_0, p_1, \ldots, p_{\log_2 t+1}$ evaluates to 1 on input z. Then $\Pr[E] \geqslant \frac{1}{3}$.*

Proof. There are three mutually-exclusive cases to consider; one of these must hold:

(1) *For all $i = 0, 1, \ldots, \log_2 t + 1$, we have $p_i(z) > 1$.* In this case, each z_j survives all stages with probability $\leqslant \frac{1}{2t}$, and so the probability that any z_j survives all stages is $\leqslant \left(\frac{1}{2t}\right) \cdot t = \frac{1}{2}$. Therefore, the probability that two or more z_j's survive all stages is $\leqslant \frac{1}{2}$. This is the same event as $p_0(z) > 1$ for all $i = 0, 1, \ldots, \log_2 t + 1$.

(2) *For all $i = 0, 1, \ldots, \log_2 t + 1$, we have $p_i(z) = 1$.* This is the event $p_0(z) = 1$.

(3) *There exists some i such that $p_i(z) > 1$ but $p_{i+1}(z) \leqslant 1$.* In this case, given a value of $p_j(z)$, we have that

$$\Pr[p_{j+1}(z) = 0] = \frac{1}{2^{p_j(z)}},$$
$$\Pr[p_{j+1}(z) = 1] = p_j(z) \cdot \frac{1}{2^{p_j(z)}}.$$

So if i is such that $p_i(z) > 1$ but $p_{i+1}(z) \leqslant 1$, then $\Pr[p_{i+1}(z) = 1] \geqslant \frac{2}{3}$, since $p_i(z) \geqslant 2$.

So since the probability of (1) is $\leqslant \frac{1}{2}$, and so the probability of one of (2) or (3) is $\geqslant \frac{1}{2}$, the probability of the event E is $\geqslant \frac{1}{2} \cdot \frac{2}{3} = \frac{1}{3}$. □

We have argued above that for any fixed input, one of the polynomials $p_0, p_1, \ldots, p_{\log_2 t+1}$ gets z right with probability $\geqslant \frac{1}{3}$. We need to improve that success probability from $\frac{1}{3}$ to $1 - \varepsilon$, and get a single polynomial. Towards this, define

$$r(x) = \prod_{i=0}^{\log(t)+1} (1 - p_i(x)).$$

Then $r(x)$ is a random polynomial of degree $\leqslant \log(t) + 1$ taking values in $\left[-t^{O(\log t)}, t^{O(\log t)}\right]$. Now if $x = 0^t$, then $r(x) = 1$ because each $p_i(x) = 0$ for all i. For any other x, some $p_i(x) = 1$ with probability $\geqslant \frac{1}{3}$ and so for such x, $r(x) = 0$ with probability $\geqslant \frac{1}{3}$ since $1 - p_i(x) = 0$ for that i.

Now we focus on improving the success probability via reducing the failure probability. Let $r'(x)$ be the product of $O\left(\log \frac{1}{\varepsilon}\right)$ many independent copies of $r(x)$. Then $r'(x)$ is a polynomial of degree $\leqslant O\left(\log \frac{1}{\varepsilon} \log t\right)$, and takes values in $\left[-t^{O(\log \frac{1}{\varepsilon}) \cdot \log t}, t^{O(\log \frac{1}{\varepsilon}) \cdot \log t}\right]$. Now for any $x \neq 0^t$, $r'(x) \neq 0$ with probability $\leqslant \left(1 - \frac{1}{3}\right)^{O\left(\log \frac{1}{\varepsilon}\right)} \leqslant \varepsilon$.

Let $a(x) = 1 - r(x)$, so that for all x (whether $x = 0^t$ or not),

$$\Pr_{\boldsymbol{a}}\left[\boldsymbol{a}(x) = \underbrace{x_1 \vee \ldots \vee x_t}_{\text{0/1-valued}}\right] \geqslant 1 - \varepsilon, \qquad (\star)$$

and so input by input, we have a random polynomial which is likely to give the right answer on x; this immediately implies that for any fixed distribution $\mathcal{D}$ over $\{0,1\}^t$, there exists an outcome of a (i.e., a polynomial) such that

$$\Pr_{\boldsymbol{x} \sim \mathcal{D}}[a(\boldsymbol{x}) = x_1 \vee \ldots \vee x_t] \geqslant 1 - \varepsilon. \qquad (\star\star)$$

We can think of this result as follows: construct the table below, and place ✓ in the cell corresponding to the input x and the polynomial a that gets x right.

a-outcomes

	$x = 0^t$	$x = \ldots$
a_1	✓	
a_2		
$\vdots$		

2^t outcomes

The guarantee ($\star$) says that for every column, at least a $1-\varepsilon$ fraction of the rows have a ✓. So for any distribution $\mathcal{D}$ over the columns, the density of ✓'s in the table (under the weighting of $\mathcal{D}$) is at least $1-\varepsilon$. So surely there must be some row such that the density of ✓'s in that row is at least $1-\varepsilon$. That row is exactly the row corresponding to the polynomial a that we want in ($\star\star$).

So we have successfully completed the proof for the case $d=1$ where $f = x_1 \vee \ldots \vee x_t$. Clearly it is also true for AND, $\neg$, etc, and so we just need to put these together. For each gate g in $f \in \mathrm{AC}^0_{s,d}$ with at most s inputs, consider the distribution $\mathcal{D}'$ of the inputs to that gate when $\boldsymbol{x} \sim \mathcal{D}$. By the $d=1$ case we just showed above, there is a degree-$O\left(\log \frac{s}{\varepsilon} \cdot \log s\right)$ polynomial which computes the value of g correctly with probability $\geqslant 1 - \frac{\varepsilon}{s}$ on the inputs to g distributed according to $\mathcal{D}'$. Now we just replace each gate by its corresponding polynomial; the resulting composition of polynomials is a polynomial $p(x)$ of degree $O\left(\log \frac{s}{\varepsilon} \cdot \log s\right)^d$ which computes f correctly with probability $\geqslant 1 - \frac{\varepsilon}{s} \cdot s = 1-\varepsilon$, by the union bound over all gates, which proves (i). For (ii), note that the degree of the polynomial is $O\left(\log \frac{s}{\varepsilon} \cdot \log s\right)^d$, and so the maximum value of the polynomial is $\exp\left(O\left(\log \frac{s}{\varepsilon} \cdot \log s\right)^d\right)$ since the polynomial is a sum of $O\left(\log \frac{s}{\varepsilon} \cdot \log s\right)^d$ many terms, each of which is at most s. Thus ends the proof. □

§9.2.2 Polynomially-approximating AC^0 better: Linial-Mansour-Nisan's theorem

This second polynomial-approximator can be thought of as an L_2-approximator, and it is due to Linial, Mansour, and Nisan [LMN93]. (By L_2, we mean the expected squared error under the uniform distribution.) The theorem statement is as follows:

Theorem 9.3 (LMN theorem, [LMN93])**.** *There exists a real polynomial p_2 of degree $O\left(\left(\log \frac{s}{\varepsilon}\right)^d\right)$ such that*

$$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}_n}\left[(f(\boldsymbol{x}) - p_2(\boldsymbol{x}))^2\right] \leqslant \varepsilon.$$

Here too, note that p_2 may not serve as a good sandwiching polynomial, since the error is only guaranteed to be small on average and may (rarely) be very large on some inputs.

Notation 9.6. Denote the Fourier weight functionals $\mathscr{W}^k(f)$ and $\mathscr{W}^{\geqslant k}(f)$ by

$$\mathscr{W}^k(f) \coloneqq \sum_{\substack{S\subseteq[n]\\|S|=k}} \widehat{f}(S)^2, \qquad \mathscr{W}^{\geqslant k}(f) \coloneqq \sum_{\substack{S\subseteq[n]\\|S|\geqslant k}} \widehat{f}(S)^2.$$

Recall that if $f\colon \{0,1\}^n \to \{\pm 1\}$ is a function, then

$$\mathscr{W}^{\geqslant 0}(f) = \sum_{S\subseteq[n]} \widehat{f}(S)^2 = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}_n}[f(\boldsymbol{x})^2] = 1,$$

and

$$\mathscr{W}^{0}(f) = \sum_{\substack{S\subseteq[n]\\|S|=0}} \widehat{f}(S)^2 = \widehat{f}(\emptyset)^2.$$

In the proof of Theorem 9.3, we will argue that for certain values of k, the Fourier tail $\mathcal{W}^{\geqslant k}(f)$ is very small, and so we can discard high-degree parities to get a low-degree polynomial which is an L_2-good-approximator for f.

To start with, note that the LMN theorem follows directly from the following Fourier concentration bound:

Theorem 9.4. *If $f \in \mathrm{AC}^0_{s,d}$, then for any $r \in \mathbb{R}$, we have*

$$\mathscr{W}^{>r}(f) \leqslant 2s \cdot 2^{-r^{1/d}/20}.$$

Proof of Theorem 9.3 given Theorem 9.4. Given Theorem 9.4 as fact (we will prove it soon), take $r = \left(20 \log \frac{2s}{\varepsilon}\right)^d$. Then

$$\mathscr{W}^{>r}(f) \leqslant 2s \cdot 2^{-r^{1/d}/20} = 2s \cdot 2^{-20 \log \frac{2s}{\varepsilon}/20} = \varepsilon,$$

and so the polynomial p_2 is merely the truncated Fourier expansion of f at level r:

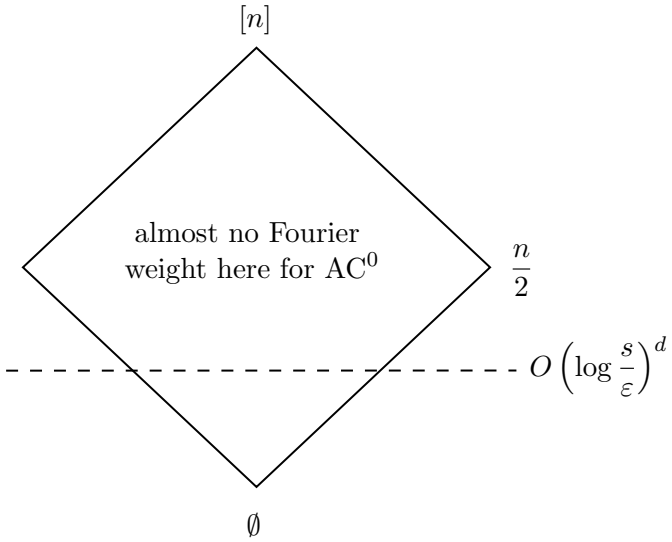
$$p_2(\boldsymbol{x}) = \sum_{\substack{S \subseteq [n] \\ |S| \leqslant r}} \widehat{f}(S) \cdot \chi_S(\boldsymbol{x}),$$

where $\chi_S(\boldsymbol{x}) = \prod_{i \in S} x_i$ for $x_i \in \{\pm 1\}$. Then by Parseval's identity, we have

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{x} \sim \mathcal{U}_n}\left[(f(\boldsymbol{x}) - p_2(\boldsymbol{x}))^2\right] &= \sum_{S \subseteq [n]} \widehat{f - p_2}(S)^2 \\
&= \sum_{\substack{S \subseteq [n] \\ |S| > r}} \left(\widehat{f}(S) - \widehat{p_2}(S)\right)^2 \\
&\leqslant \sum_{\substack{S \subseteq [n] \\ |S| > r}} \left(\widehat{f}(S)\right)^2 \\
&\leqslant \mathscr{W}^{>r}(f) \leqslant \varepsilon,
\end{aligned}$$

by our choice of r. □

So we have shown that most of the Fourier weight of an AC^0 function is concentrated in low-degree Fourier coefficients, below $\left(20 \log \frac{2s}{\varepsilon}\right)^d$. The following cartoon illustrates this idea.



Our goal is now to prove Theorem 9.4. We will do this by means of the following two main steps:

- Use the switching lemma to show that the Fourier weight of a function is concentrated on low-degree Fourier coefficients.
- Present a useful combination of random restrictions and Fourier analysis to show that the Fourier weight of an AC^0 function is concentrated on sufficiently-low-degree Fourier coefficients.

We have already seen the first step:

Lemma 9.7. *Let $f \in \mathrm{AC}^0_{s,d}$. Fix t, and let $\rho \sim \mathbb{R}_p$ where $*$-probability is $p \leqslant \left(10^d \cdot t^{d-1}\right)^{-1}$. Then*

$$\Pr_{\rho \sim \mathbb{R}_p}\left[\text{DT-depth}\left(f\restriction_\rho\right) \geqslant t\right] \leqslant s \cdot 2^{-d}.$$

Proof. This is the exact same proof as that of Lemma 3.9, modulo some factors and changes in notation. □

The second is a generic statement about how the Fourier spectrum behaves when functions are hit with a random restriction.

Lemma 9.8. *For any $f\colon \{\pm 1\}^n \to \{\pm 1\}$ and any $p \leqslant \frac{1}{10}$, we have*

$$\mathscr{W}^{\geqslant \frac{t}{p}}(f) \leqslant 2 \cdot \mathbb{E}_{(\boldsymbol{J},\boldsymbol{z}) \sim \mathcal{R}_p}\left[\mathscr{W}^{\geqslant t}\left(f_{\boldsymbol{J},\boldsymbol{z}}\right)\right].$$

For this lemma we're merely presenting a more explicit way of talking about the restriction. In particular, we can view "$(\boldsymbol{J},\boldsymbol{z}) \sim \mathcal{R}_p$" as the exact same as "$\rho \sim \mathcal{R}_p$", where the $\boldsymbol{J} \subseteq [n]$ is the set of $*$-variables and $\boldsymbol{z} \in \{\pm 1\}^{|\overline{J}|}$ is the assignment to the non-$*$-variables.

In our typical backtracking style, we first give a proof of Theorem 9.4 assuming Lemmas 9.7 and 9.8, and then we will prove Lemma 9.8 next time.

Proof of Theorem 9.4 given Lemmas 9.7 and 9.8. Take

$$p = \frac{1}{10 \cdot r^{(d-1)/d}} \implies pr = \frac{r^{1/d}}{10}.$$

Let $t = \frac{1}{2}pr = \frac{1}{20}r^{1/d}$, so that

$$\frac{1}{10^d \cdot t^{d-1}} = \frac{20^{d-1}}{10^d \cdot r^{\frac{d-1}{d}}} = \frac{2^{d-1}}{10 r^{(d-1)/d}} = 2^{d-1} \cdot p \geqslant p.$$

So we can use Lemma 9.7 with t and p, and get that

$$\Pr_{\rho \sim \mathbb{R}_p}\left[\text{DT-depth}\left(f\restriction_\rho\right) \geqslant t\right] \leqslant s \cdot 2^{-t}.$$

This means that

$$\mathbb{E}_{(\boldsymbol{J},\boldsymbol{z}) \sim \mathcal{R}_p}\left[\mathscr{W}^{\geqslant t}\left(f_{\boldsymbol{J},\boldsymbol{z}}\right)\right] \leqslant s \cdot 2^{-t},$$

since a decision tree of depth $t-1$ has no Fourier weight above t, and so we can bound the expected Fourier weight after the random restriction above level t as being at most the failure probability $s \cdot 2^{-t}$. Then by Lemma 9.8 with $t = \frac{1}{20} r^{1/d}$ and $\frac{t}{p} = \frac{r}{2}$,

$$\begin{aligned} \mathscr{W}^{\geqslant \frac{r}{2}}(f) &\leqslant 2 \cdot \mathbb{E}_{(\boldsymbol{J},\boldsymbol{z})\sim\mathcal{R}_p}\left[\mathscr{W}^{\geqslant t}\left(f_{\boldsymbol{J},\boldsymbol{z}}\right)\right] \\ &\leqslant 2s \cdot 2^{-t} = 2s \cdot 2^{-\frac{r^{1/d}}{20}}, \end{aligned}$$

and since we have the extra slack from $\frac{r}{2}$ to r, the claim of Theorem 9.4 follows. □

Next time, we will prove Lemma 9.8, and then we will be able to complete the proof of Theorem 9.3.

§10 Lecture 10—26th March, 2024

Last time. In the previous class, we:

- Finished the proof of Viola's theorem.
- Introduced Braverman's theorem [Bra08]: $(\log n)^{O(1)}$-wise independence fools AC^0 circuits. Some tools towards this:
  - Beigel-Reingold-Spielman's theorem [BRS91]: AC^0 circuits can be approximated by low-degree polynomials.
  - Linial-Mansour-Nisan's theorem [LMN93]: AC^0 circuits can be approximated by low-degree polynomials with small Fourier mass. Started proving this.

Today. We will:

- Finish the proof of Linial-Mansour-Nisan's theorem (the result about Fourier weight under restrictions).
- Proof of Braverman's theorem combining the Beigel-Reingold-Spielman and Linial-Mansour-Nisan results.
- Start a new unit on linear threshold functions (LTFs): basics, examples, regularity, the Berry-Esseen theorem.
- Start a sketch of a PRG for LTFs: $\widetilde{O}\left(1/\varepsilon^2\right)$-wise independence ε-fools LTFs.

Reminder: the project progress report is due Thursday April 4th.

---

§10.1 Finishing the proof of Linial-Mansour-Nisan's theorem

Recall from last time that we needed to prove the following lemma to complete the proof of Theorem 9.3:

Lemma 10.1. *For any Boolean function $f\colon \{\pm 1\}^n \to \{\pm 1\}$ and any $p \leqslant \frac{1}{10}$, we have*

$$\mathscr{W}^{\geqslant \frac{t}{p}}(f) \leqslant 2 \cdot \mathbb{E}_{(\boldsymbol{J},\boldsymbol{z})\sim\mathcal{R}_p}\left[\mathscr{W}^{\geqslant t}\left(f_{\boldsymbol{J},\boldsymbol{z}}\right)\right].$$

To prove this result, here are the steps:

Claim 10.2. *For any Boolean function $f\colon \{\pm 1\}^n \to \{\pm 1\}$ and any $S \subseteq [n]$, we have*

$$\widehat{f_{J,z}}(S) = \begin{cases} 0 & \text{if } S \not\subseteq J, \\ \sum_{T \subseteq \overline{J}} \widehat{f}(S \cup T) \cdot \chi_T(z) & \text{if } S \subseteq J, \end{cases}$$

where $\overline{J} = [n] \setminus J$.

Claim 10.3. *For fixed $J, S \subseteq [n]$ and a uniformly random $\boldsymbol{z} \in \{\pm 1\}^{\overline{J}}$,*

$$\mathbb{E}_{\boldsymbol{z} \in \{\pm 1\}^{\overline{J}}}\left[\widehat{f_{J,\boldsymbol{z}}}(S)\right] = \mathbb{1}\{S \subseteq J\} \cdot \widehat{f}(S)$$
$$\mathbb{E}_{\boldsymbol{z} \in \{\pm 1\}^{\overline{J}}}\left[\widehat{f_{J,\boldsymbol{z}}}(S)^2\right] = \mathbb{1}\{S \subseteq J\} \cdot \sum_{T \subseteq \overline{J}} \widehat{f}(S \cup T)^2.$$

Claim 10.4. *For $(\boldsymbol{J}, \boldsymbol{z}) \sim \mathcal{R}_p$ and any $S \subseteq [n]$, we have*

$$\mathbb{E}_{(\boldsymbol{J},\boldsymbol{z}) \sim \mathcal{R}_p}\left[\widehat{f_{\boldsymbol{J},\boldsymbol{z}}}(S)\right] = p^{|S|} \cdot \widehat{f}(S)$$
$$\mathbb{E}_{(\boldsymbol{J},\boldsymbol{z}) \sim \mathcal{R}_p}\left[\widehat{f_{\boldsymbol{J},\boldsymbol{z}}}(S)^2\right] = \sum_{U \subseteq [n]} \widehat{f}(U)^2 \cdot \Pr_{\boldsymbol{J}}[U \cap \boldsymbol{J} = S].$$

Claim 10.5. *The expectation of the Fourier tail weight of $f_{\boldsymbol{J},\boldsymbol{z}}$, under the distribution $\mathcal{R}_p$ and above some level k is*

$$\mathbb{E}_{(\boldsymbol{J},\boldsymbol{z}) \sim \mathcal{R}_p}\left[\mathscr{W}^{\geqslant k}\left(f_{\boldsymbol{J},\boldsymbol{z}}\right)\right] = \sum_{r \geqslant k} \mathscr{W}^{r}(f) \cdot \Pr[\mathsf{Bin}(r,p) \geqslant k].$$

Now we show that these claims (particularly the last) imply Lemma 10.1.

Proof of Lemma 10.1 given Claim 10.5. Observe that Claim 10.5 gives

$$\begin{aligned} \mathbb{E}_{(\boldsymbol{J},\boldsymbol{z}) \sim \mathcal{R}_p}\left[\mathscr{W}^{\geqslant k}\left(f_{\boldsymbol{J},\boldsymbol{z}}\right)\right] &= \sum_{r \geqslant k} \mathscr{W}^{r}(f) \cdot \Pr[\mathsf{Bin}(r,p) \geqslant k] \\ &\geqslant \sum_{r \geqslant k/p} \mathscr{W}^{r}(f) \cdot \Pr[\mathsf{Bin}(r,p) \geqslant k], \end{aligned}$$

since $k \leqslant k/p$ for $p \leqslant 1$. For each $r \geqslant k/p$, we have $\Pr[\mathsf{Bin}(r,p) \geqslant k] \geqslant \frac{1}{2}$, since the probability that the binomial distribution exceeds its mean is at least $\frac{1}{2}$. So

$$\begin{aligned} \sum_{r \geqslant k/p} \mathscr{W}^{r}(f) \cdot \Pr[\mathsf{Bin}(r,p) \geqslant k] &\geqslant \frac{1}{2} \cdot \sum_{r \geqslant k/p} \mathscr{W}^{r}(f) \\ &= \frac{1}{2} \cdot \mathscr{W}^{\geqslant k/p}(f), \end{aligned}$$

which is exactly what Lemma 10.1 states with $t = k$. This completes the proof. □

Now we prove these claims.

Proof of Claim 10.2. View $f_{J,z}$ as an n-variable function that only depends on the variables in J—a J-junta—in the sense that $f_{J,z}(x) = f(x_J, z)$ for all $x \in \{\pm 1\}^n$, and suppose $S \not\subseteq J$, so that there is some variable $x_i \in S$ with $i \notin J$. S contains an irrelevant variable for $f_{J,z}$, so the correlation between $f_{J,z}$ and S is zero. This correlation is exactly the Fourier coefficient $\widehat{f_{J,z}}(S)$, so $\widehat{f_{J,z}}(S) = 0$.

For the second case, if $S \subseteq J$, with $f_{J,z}(x) = f(x_J, z)$, we can write

$$\begin{aligned} f_{J,z}(x_J, z) &= \sum_{R \subseteq [n]} \widehat{f}(R) \cdot \chi_R(x_J, z) = \sum_{S \subseteq J} \sum_{T \subseteq \overline{J}} \widehat{f}(S \cup T) \cdot \chi_S(x_J) \cdot \chi_T(z) \\ &= \sum_{S \subseteq T} \chi_S(x_J) \cdot \sum_{T \subseteq \overline{J}} \widehat{f}(S \cup T) \cdot \chi_T(z), \end{aligned}$$

where the last equality follows from the fact that $\chi_S(x_J)$ is orthogonal to $\chi_{S'}(x_J)$ for $S \neq S'$. The Fourier coefficient $\widehat{f_{J,z}}(S)$ is the coefficient of $\chi_S(x_J)$ in the above expression, which is exactly $\sum_{T \subseteq \overline{J}} \widehat{f}(S \cup T) \cdot \chi_T(z)$. □

Proof of Claim 10.3. We have

$$\begin{aligned} \mathbb{E}_{\boldsymbol{z} \in \{\pm 1\}^{\overline{J}}} \left[\widehat{f_{J,\boldsymbol{z}}}(S) \right] &= \mathbb{E}_{\boldsymbol{z} \in \{\pm 1\}^{\overline{J}}} \left[\sum_{T \subseteq \overline{J}} \widehat{f}(S \cup T) \cdot \chi_T(z) \right] \\ &= \sum_{T \subseteq \overline{J}} \widehat{f}(S \cup T) \cdot \mathbb{E}_{\boldsymbol{z} \in \{\pm 1\}^{\overline{J}}} \left[\chi_T(z) \right]. \end{aligned}$$

The expectation $\mathbb{E}_{\boldsymbol{z} \in \{\pm 1\}^{\overline{J}}} [\chi_T(z)]$ is 0 if $T \neq \emptyset$ and 1 if $T = \emptyset$, so we get that

$$\mathbb{E}_{\boldsymbol{z} \in \{\pm 1\}^{\overline{J}}} \left[\widehat{f_{J,\boldsymbol{z}}}(S) \right] = \mathbb{1}\{S \subseteq J\} \cdot \widehat{f}(S).$$

For the second part, we have

$$\begin{aligned} \mathbb{E}_{\boldsymbol{z} \in \{\pm 1\}^{\overline{J}}} \left[\widehat{f_{J,\boldsymbol{z}}}(S)^2 \right] &= \mathbb{E}_{\boldsymbol{z} \in \{\pm 1\}^{\overline{J}}} \left[\left(\sum_{T \subseteq \overline{J}} \widehat{f}(S \cup T) \cdot \chi_T(z) \right)^2 \right] \\ &= \sum_{T \subseteq \overline{J}} \widehat{f}(S \cup T)^2 \cdot \mathbb{E}_{\boldsymbol{z} \in \{\pm 1\}^{\overline{J}}} \left[\chi_T(z)^2 \right]. \end{aligned}$$

The expectation $\mathbb{E}_{\boldsymbol{z} \in \{\pm 1\}^{\overline{J}}} \left[\chi_T(z)^2 \right]$ is 0 if $T \neq \emptyset$ and 1 if $T = \emptyset$, so we get that

$$\mathbb{E}_{\boldsymbol{z} \in \{\pm 1\}^{\overline{J}}} \left[\widehat{f_{J,\boldsymbol{z}}}(S)^2 \right] = \mathbb{1}\{S \subseteq J\} \cdot \sum_{T \subseteq \overline{J}} \widehat{f}(S \cup T)^2.$$ □

Proof of Claim 10.4. Proceeding straightforwardly, we have

$$\begin{aligned} \mathbb{E}_{(\boldsymbol{J}, \boldsymbol{z}) \sim \mathcal{R}_p} \left[\widehat{f_{\boldsymbol{J},\boldsymbol{z}}}(S) \right] &= \sum_{J \subseteq [n]} \Pr_{\boldsymbol{J}}[J = \boldsymbol{J}] \cdot \mathbb{E}_{\boldsymbol{z} \in \{\pm 1\}^{\overline{J}}} \left[\widehat{f_{J,\boldsymbol{z}}}(S) \right] \\ &= \sum_{J \subseteq [n]} p^{|J|} \cdot \mathbb{1}\{S \subseteq J\} \cdot \widehat{f}(S) \\ &= p^{|S|} \cdot \widehat{f}(S). \end{aligned}$$

For the second part, we have

$$\begin{aligned}\mathbb{E}_{(\boldsymbol{J},\boldsymbol{z})\sim\mathcal{R}_p}\left[\widehat{f_{\boldsymbol{J},\boldsymbol{z}}}(S)^2\right] &= \sum_{J\subseteq[n]} \Pr_{\boldsymbol{J}}[J=\boldsymbol{J}]\cdot \mathbb{E}_{\boldsymbol{z}\in\{\pm1\}^{\overline{J}}}\left[\widehat{f_{J,\boldsymbol{z}}}(S)^2\right] \\ &= \sum_{J\subseteq[n]} p^{|J|}\cdot \mathbb{1}\{S\subseteq J\}\cdot \sum_{T\subseteq\overline{J}} \widehat{f}(S\cup T)^2 \\ &= \sum_{U\subseteq[n]} \widehat{f}(U)^2\cdot \Pr_{\boldsymbol{J}}[U\cap\boldsymbol{J}=S]. \qquad \square\end{aligned}$$

Proof of Claim 10.5. We have

$$\begin{aligned}\mathbb{E}_{(\boldsymbol{J},\boldsymbol{z})\sim\mathcal{R}_p}\left[\mathscr{W}^{\geqslant k}(f_{\boldsymbol{J},\boldsymbol{z}})\right] &= \sum_{r\geqslant k} \mathscr{W}^r(f)\cdot \Pr[\mathsf{Bin}(r,p)\geqslant k] \\ &= \sum_{r\geqslant k}\sum_{S\subseteq[n]} \widehat{f}(S)^2\cdot \Pr[\mathsf{Bin}(r,p)\geqslant k]\cdot \mathbb{1}\{S\subseteq\boldsymbol{J}\} \\ &= \sum_{S\subseteq[n]} \widehat{f}(S)^2\cdot \sum_{r\geqslant k} \Pr[\mathsf{Bin}(r,p)\geqslant k]\cdot \mathbb{1}\{S\subseteq\boldsymbol{J}\}.\end{aligned}$$

The inner sum is $\Pr[\mathsf{Bin}(r,p)\geqslant k] = \sum_{r\geqslant k}\binom{r}{k}p^k(1-p)^{r-k}$, which is the probability that a binomial random variable exceeds k. This is at least $\Pr[\mathsf{Bin}(r,p)\geqslant k]\geqslant \Pr[\mathsf{Bin}(r,p)=k] = \binom{r}{k}p^k(1-p)^{r-k}$, so

$$\sum_{r\geqslant k}\Pr[\mathsf{Bin}(r,p)\geqslant k]\geqslant \sum_{r\geqslant k}\Pr[\mathsf{Bin}(r,p)=k] = 1.$$

Therefore, we have

$$\begin{aligned}\mathbb{E}_{(\boldsymbol{J},\boldsymbol{z})\sim\mathcal{R}_p}\left[\mathscr{W}^{\geqslant k}(f_{\boldsymbol{J},\boldsymbol{z}})\right] &= \sum_{S\subseteq[n]} \widehat{f}(S)^2\cdot \sum_{r\geqslant k}\Pr[\mathsf{Bin}(r,p)\geqslant k]\cdot \mathbb{1}\{S\subseteq\boldsymbol{J}\} \\ &\leqslant \sum_{r\geqslant k}\mathscr{W}^r(f)\cdot \Pr[\mathsf{Bin}(r,p)\geqslant k]. \qquad \square\end{aligned}$$

§10.2 Proof of Braverman's theorem

We have now proved the LMN approximator, and we have the following arrows in our quiver:

Lemma 10.6 (BRS theorem, [BRS91])**.** *Let $f\in \mathrm{AC}^0_{s,d}$. Let $\mathcal{D}$ be any distribution over $\{0,1\}^n$. Then there exists a real polynomial p such that*

$$\Pr_{\boldsymbol{x}\sim\mathcal{D}}[p(\boldsymbol{x})=f(\boldsymbol{x})]\geqslant 1-\varepsilon,$$

where

(i) $\deg(p)\leqslant \left(\log\frac{s}{\varepsilon}\right)^{O(d)}$*, and*

(ii) for every input $x\in\{0,1\}^n$, $|p(x)|\leqslant \exp\left(\left(\log\frac{s}{\varepsilon}\right)^{O(d)}\right)$.

Theorem 10.1 (LMN theorem, [LMN93])**.** *There exists a real polynomial p_2 of degree $O\left(\left(\log \frac{s}{\varepsilon}\right)^d\right)$ such that*

$$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}_n}\left[(f(\boldsymbol{x})-p_2(\boldsymbol{x}))^2\right] \leqslant \varepsilon.$$

Now we prove Braverman's theorem. First, more information about the approximation in Lemma 10.6. We will augument that lemma to get that for any distribution $\mathcal{D}$ over $\{0,1\}^n$, there exists a low-degree polynomial that gets it right almost everywhere; such a polynomial also gets it wrong somewhere, and it turns out via our augumentation that (modulo some technicalities) there is an AC^0 circuit which recognises when an input is in the error region of the polynomial.

Lemma 10.7 (Extended BRS)**.** *Let $f \in \mathrm{AC}^0_{s,d}$. Let $\mathcal{D}$ be any distribution over $\{0,1\}^n$. Then there exists a real polynomial p such that*

$$\Pr_{\boldsymbol{x}\sim\mathcal{D}}[p(\boldsymbol{x}) = f(\boldsymbol{x})] \geqslant 1-\varepsilon,$$

and an "error-recognising" circuit E of size $\mathrm{poly}(s)$ and depth $d+O(1)$ where

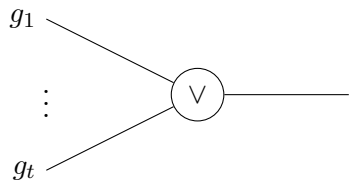
(i) $\deg(p) \leqslant \left(\log \frac{s}{\varepsilon}\right)^{O(d)}$*, and*

(ii) for every input $x \in \{0,1\}^n$, $|p(x)| \leqslant \exp\left(\left(\log \frac{s}{\varepsilon}\right)^{O(d)}\right)$.

(iii) if $E(x)=0$, then $p(x)=f(x)$ (that is, $p(x)\neq f(x)$ implies $E(x)=1$), and E doesn't fire too often, i.e. $\Pr_{\boldsymbol{x}\sim\mathcal{D}}[E(\boldsymbol{x})=1] \leqslant \varepsilon$.

Proof. We only need to see the proof for the claim about the error-recognising circuit E, since the rest of the lemma follows from the proof of Lemma 9.4. The idea for the proof is as follows: E checks whether anything went wrong in any of the intermediate stages in building the polynomial p that approximates f, and if so, it outputs 1, and so we need to describe E and show that it is in $\mathrm{AC}^0_{\mathrm{poly}(s),d+O(1)}$.

Consider any fixed OR gate in the circuit for f, with inputs $g_1, g_2, \ldots, g_t$ for some $t \leqslant s$.



In the polynomial, we used

$$1 - \prod_{i=1}^{\mathrm{polylog}(t/\varepsilon)} \left(1 - \sum_{j\in S_i} g_j\right)$$

for different sets $S_i \subseteq [t]$. If the polynomial is wrong, then exactly these two things happen:

- at least one of $g_1, g_2, \ldots, g_s$ is 1, but
- each set S_i has either no 1's or $\geqslant 2$ of the 1's in $g_1, \ldots, g_t$.

This is checkable in constant depth; for example, the first criterion can be checked by the depth-1 circuit

$$\bigvee_{i=1}^{t} g_i,$$

and the second criterion can be checked by the depth-2 circuit

$$\bigvee_{1 \leqslant a < b \leqslant t} g_a \wedge g_b.$$

Moreover, we can do this in parallel for each OR gate in the circuit for f, and then take the OR of all these checks to get E. If any of these checks fails, then E outputs 1; otherwise, it outputs 0 (and $p(x) = f(x)$). The size of E is $\mathrm{poly}(s)$ and the depth is $d + O(1)$, since we are doing a constant amount of work for each OR gate in the circuit for f with a constant-factor blowup in depth for the parallelism. Furthermore, we know by the argument in Lemma 9.4 that the probability that E outputs 1 is at most ε. This completes the proof. □

Now that we have gotten more information about the BRS approximator, we will present a refined goal for sandwiching. Recall the statement of Corollary 8.12:

Corollary 10.8 (Sandwiching polynomials give PRGs)**.** *A function $f\colon \{0,1\}^n \to \mathbb{R}$ is ε-fooled by any k-wise independent distribution $\boldsymbol{X}$ if there exist ε-sandwiching real polynomials q_ℓ, q_u of degree k such that $q_\ell(x) \leqslant f(x) \leqslant q_u(x)$ and $\mathbb{E}_{\boldsymbol{x}\sim\boldsymbol{X}}[q_u(\boldsymbol{x}) - q_\ell(\boldsymbol{x})] \leqslant \varepsilon$.*

Observe that since AC^0 is closed under negation, to give q_u and q_ℓ enough power to sandwich AC^0, it is enough to just give q_ℓ such that $q_\ell(x) \leqslant f(x)$ and $\mathbb{E}_{\boldsymbol{x}\sim\boldsymbol{X}}[f(\boldsymbol{x}) - q_\ell(\boldsymbol{x})] \leqslant \varepsilon/2$, in which case $q_u = 1 - q_\ell$ will sandwich f from above.

A bigger observation is that it is enough to give a lower sandwiching polynomial q_ℓ which has the following two properties:

(i) $q_\ell(x)$ can depend on the particular k-wise independent distribution, say $\mathcal{D}$, and

(ii) q_ℓ lower-sandwiches f', where f' is close to f both under $\mathcal{D}$ and under the uniform distribution $\mathcal{U}_n$.

We will now prove a lemma that says that given a good-enough approximator for f' under $\mathcal{D}$ and $\mathcal{U}_n$ and we have a lower-sandwicher q_ℓ for f', then f and f' are close under $\mathcal{D}$.

Lemma 10.9. *Suppose that for every k-wise independent distribution $\mathcal{D}$, we have a Boolean function $f'\colon \{0,1\}^n \to \{0,1\}$ and a degree-k polynomial q_ℓ such that*

(i) $\Pr_{\boldsymbol{x}\sim\mathcal{D}}[f(\boldsymbol{x}) \neq f'(\boldsymbol{x})] \leqslant \frac{\varepsilon}{3}$ *and* $\Pr_{\boldsymbol{x}\sim\mathcal{U}_n}[f(\boldsymbol{x}) \neq f'(\boldsymbol{x})] \leqslant \frac{\varepsilon}{3}$*, and*

(ii) q_ℓ is a lower-sandwicher for f', that is $q_\ell \leqslant f'$ and $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}_n}[f'(\boldsymbol{x}) - q_\ell(\boldsymbol{x})] \leqslant \frac{\varepsilon}{3}$.

Then for every k-wise independent distribution $\mathcal{D}$, we have

$$\mathbb{E}_{\boldsymbol{u}\sim\mathcal{U}_n}[f(\boldsymbol{u})] - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[f(\boldsymbol{x})] \leqslant \varepsilon.$$

Note that combining this with the q_u version of the statement of Lemma 10.9 gives that f is ε-fooled by the k-wise independent distribution $\mathcal{D}$.

Proof of Lemma 10.9. The expectation of f on $\mathcal{D}$ is

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[f(\boldsymbol{x})\right] &\geqslant \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[f'(\boldsymbol{x})\right] - \frac{\varepsilon}{3} && \text{by (i)}\\
&\geqslant \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[q_\ell(\boldsymbol{x})\right] - \frac{\varepsilon}{3} && \text{since } q_\ell \leqslant f'\\
&= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}_n}\left[q_\ell(\boldsymbol{x})\right] - \frac{\varepsilon}{3} && \text{by } k\text{-wise indep. with } \deg(q_\ell) \leqslant k\\
&\geqslant \mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}_n}\left[f'(\boldsymbol{x})\right] - \frac{2\varepsilon}{3} && \text{by (ii)}\\
&\geqslant \mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}_n}\left[f(\boldsymbol{x})\right] - \varepsilon,
\end{aligned}$$

which is what we wanted to show. □

This lemma shows that all we need to do to fool AC^0 circuits is to show that the original AC^0 circuit has some f' that is close to it under both $\mathcal{D}$ and $\mathcal{U}_n$, and that we have a lower-sandwicher q_ℓ of low degree for f' under $\mathcal{D}$ and $\mathcal{U}_n$. So it suffices to show the following:

Lemma 10.10. *Let $f \in \mathrm{AC}^0_{s,d}$, let $k = \left(O\left(\log \frac{s}{\varepsilon}\right)\right)^{O(d^2)}$, and let $\mathcal{D}$ be any k-wise independent distribution. Then there exists a Boolean function f' and a degree-k polynomial $q_\ell = p$ such that (i) and (ii) of Lemma 10.9 both hold.*

To prove this result, we will start off by applying the BRS approximation to f using a clever distribution that is a mixture of the uniform distribution $\mathcal{U}_n$ and the k-wise independent distribution $\mathcal{D}$—call this distribution $\mathcal{D}' = \frac{1}{2}(\mathcal{U}_n + \mathcal{D})$—and use error parameter $\varepsilon/8$, and by so doing we will then get a polynomial p_0 such that

$$\begin{aligned}
\Pr_{\boldsymbol{x}\sim\mathcal{D}'}\left[p_0(\boldsymbol{x}) \neq f(\boldsymbol{x})\right] &\leqslant \frac{\varepsilon}{4},\\
\Pr_{\boldsymbol{x}\sim\mathcal{U}_n}\left[p_0(\boldsymbol{x}) \neq f(\boldsymbol{x})\right] &\leqslant \frac{\varepsilon}{4},
\end{aligned}$$

as well as a circuit E with $\mathrm{poly}(s)$ size and depth $d + O(1)$ that recognises when p_0 is wrong with probability at most $\frac{\varepsilon}{4}$. Next, we apply the LMN approximation on E in the following sense: let $p_{E,2}$ be the polynomial of degree $O\left(\left(\log \frac{s}{\varepsilon}\right)^{O(d)}\right)$ such that

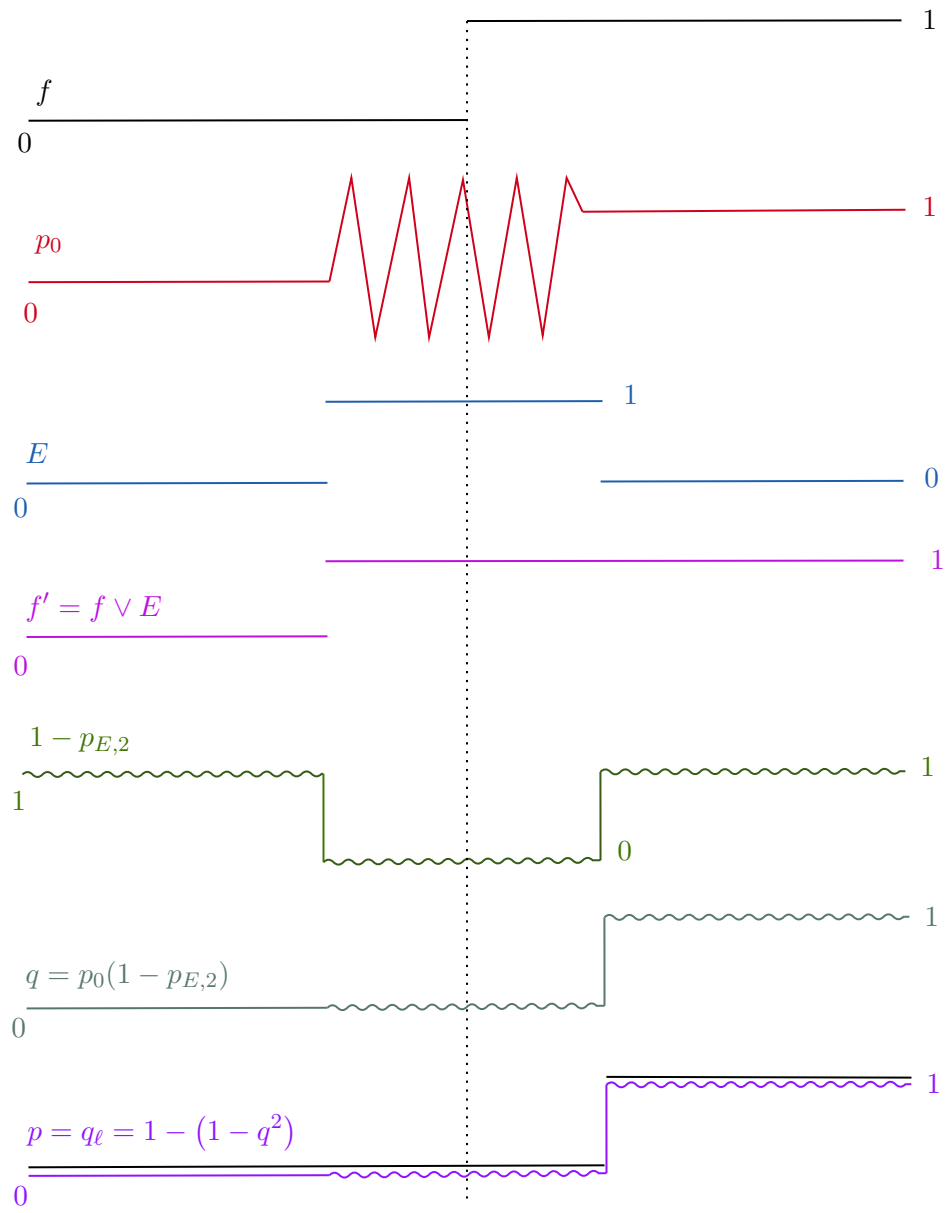
$$\mathbb{E}_{\boldsymbol{u}\sim\mathcal{U}}\left[\left(E(\boldsymbol{u}) - p_{E,2}(\boldsymbol{u})\right)^2\right] \leqslant \delta,$$

where we will choose δ appropriately later. We then set $f' = f \vee E$, $q = p_0\left(1 - p_{E,2}\right)$, and the final $p = q_\ell = 1 - (1-q)^2$.

But before that, why might Lemma 10.10 be true? Well:

- f' is close to f under both $\mathcal{D}$ and $\mathcal{U}_n$ because E is almost 0 everywhere under both $\mathcal{D}$ and $\mathcal{U}_n$.
- p_0 may make wild errors on f' when $E(x) = 1$; multiplying by $1 - p_{E,2}$ "tames" these errors, damping them to something we can handle. But $q = p_0\left(1 - p_{E,2}\right)$ may not be a lower-sandwicher when $f' = 1$.
- Taking $p = 1 - (1-q)^2$ gives us a lower-sandwicher for f', as it is always $\leqslant 1$.

The following picture may clarify the situation better (the non-straight lines represent the error regions—proportional to their non-straightness—of the functions, polynomials, and circuits):



Let us now see the proof details.

Proof of Lemma 10.10. We need to verify (i) and (ii) in Lemma 10.9. The first part, (i) is easy, because $f' \neq f$ only if $E = 1$ and under $\mathcal{D}$ or $\mathcal{U}_n$, we have that $\Pr[E = 1] \leqslant \frac{\varepsilon}{4}$, and so we're done. For the second part, we need to show that $p = q_\ell = 1 - (1 - q^2)$ is a lower-sandwicher for f'; we do this by means of two claims.

Claim 10.11. *If $f'(x) = 0$, then $q(x) = p_0(x)(1 - p_{E,2}) = 0$.*

Proof. If $f' = 0$, then $E = 0$—since $f' = f \vee E$—and we are not in the error region. Thus p_0 agrees with f, so that $p_0(x) = f(x) = 0$, and so $q(x) = 0$. □

Now recall the metric $\|\cdot\|_2$ on functions $a, b\colon \{\pm 1\}^n \to \mathbb{R}$: $\|a - b\|_2 = \sqrt{\mathbb{E}_{\boldsymbol{u} \sim \mathcal{U}_n}[(a(\boldsymbol{u}) - b(\boldsymbol{u}))^2]}$.

Claim 10.12. *The error*

$$\|f' - q\|_2 \leqslant \sqrt{\frac{\varepsilon}{4}} + \exp\left(\log\left(\frac{s}{\varepsilon}\right)\right)^{O(d)} \cdot \sqrt{\delta} \leqslant \sqrt{\frac{\varepsilon}{3}}$$

by taking $\delta = \varepsilon \cdot \exp\left(-\log\left(\frac{s}{\varepsilon}\right)\right)^{O(d)}$.

Proof. The triangle inequality gives us

$$\|f' - q\|_2 \leqslant \|f' - p_0(1-E)\|_2 + \|p_0(1-E) - q\|_2.$$

Now observe that

$$\|f' - p_0(1-E)\|_2 \leqslant \sqrt{\Pr_{\mathcal{U}}[E=1]} \leqslant \sqrt{\frac{\varepsilon}{4}}.$$

For the second, write $p_0(1-E) - q = p_0(p_{E,2} - E)$. Using our pointwise bound on p_0, we know that $\max_{x\in\{0,1\}^n} |p_0(x)| \leqslant \exp\left(\log\left(\frac{s}{\varepsilon}\right)^{O(d)}\right)$, and so

$$\begin{aligned}\|p_0(p_{E,2} - E)\|_2 &\leqslant \exp\left(\log\left(\frac{s}{\varepsilon}\right)^{O(d)}\right) \cdot \|p_{E,2} - E\|_2 \\ &\leqslant \exp\left(\log\left(\frac{s}{\varepsilon}\right)^{O(d)}\right) \cdot \sqrt{\delta},\end{aligned}$$

by our LMN-sense approximation of $p_{E,2}$ by E. The proof is essentially done; we can straightforwardly set $\delta = \varepsilon \cdot \exp\left(-\log\left(\frac{s}{\varepsilon}\right)\right)^{O(d)}$ to get the desired bound. □

Now we have the tools we need to argue part (ii) of Lemma 10.9 for Lemma 10.10. We have that $p \leqslant f'$ pointwise, because if $f' = 0$, then $q = 0$ by Claim 10.11, and so $p = 0 \leqslant 0 = f'$. Otherwise, we have that $f' = 1$, in which case $f'(x) - p(x) = (1 - q(x))^2 = (f'(x) - q(x))^2$. So,

$$\mathbb{E}_{\boldsymbol{u}\sim\mathcal{U}_n}[(f'(\boldsymbol{u}) - p(\boldsymbol{u}))] = \mathbb{E}_{\boldsymbol{u}\sim\mathcal{U}_n}[\left|f'(\boldsymbol{u}) - p(\boldsymbol{u})\right|] \leqslant \mathbb{E}_{\boldsymbol{u}\sim\mathcal{U}_n}[\left(f'(\boldsymbol{u}) - q(\boldsymbol{u})\right)^2] \leqslant \frac{\varepsilon}{3},$$

where the last inequality follows from Claim 10.12. To wrap things up we will show that the degree is what we claimed it was. Indeed,

$$\deg(p) \leqslant 2 \cdot \left(\underbrace{\deg(p_0)}_{\log\left(\frac{s}{\varepsilon}\right)^{O(d)} \text{ by BRS}} + \underbrace{\deg(p_{E,2})}_{\log\left(\frac{s}{\delta}\right)^{O(d)} \text{ by LMN}} \right) = \left(\left(\log\left(\frac{s}{\varepsilon}\right)\right)^{O(d)}\right)^{O(d)} = \left(\log\left(\frac{s}{\varepsilon}\right)\right)^{O(d^2)},$$

as claimed. □

The proof of Braverman's theorem (Theorem 9.2) is now complete.

§10.3 Linear threshold functions: basics

Linear threshold functions (LTFs) are a natural class of functions to consider in the context of complexity and learning theory. They are functions that split the n-dimensional Boolean hypercube into two regions, one of which is mapped to 1 and the other to 0. Formally, they are defined as follows.

Definition 10.13 (Linear threshold function)**.** *A function* $f\colon \{\pm 1\}^n \to \{\pm 1\}$ *is a* linear threshold function *if there exists a weight vector* $w \in \mathbb{R}^n$ *and a threshold* $\theta \in \mathbb{R}$ *such that* $f(x) = \operatorname{sign}(w \cdot x - \theta)$.

There are natural LTFs for many classes of Boolean functions. For example, for the class of OR functions over n variables,

$$x_1 \vee \ldots \vee x_n = \textsc{or}(x) = \operatorname{sign}\left(x_1 + \ldots + x_n - \frac{1}{2}\right),$$

where the weight vector is $w = (1, 1, \ldots, 1)$ and the threshold is $\theta = \frac{1}{2}$. Similarly, for the majority function, we have for $x \in \{\pm 1\}^n$,

$$\textsc{maj}(x) = \operatorname{sign}\left(\sum_{i=1}^{n} x_i\right),$$

where the weight vector is $w = (1, 1, \ldots, 1)$ and the threshold is 0. For the decision list function,

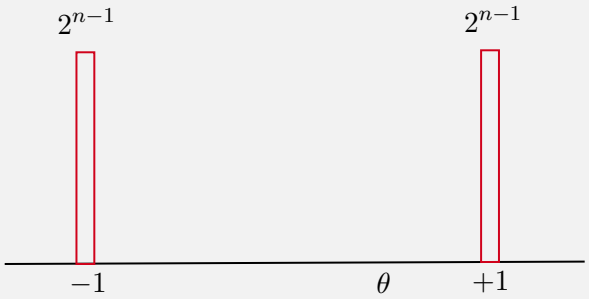
$$\textsc{dl}(x) = \operatorname{sign}\left(2^n x_1 - 2^{n-1} x_2 + \ldots + 2x_{n-1} - x_n\right),$$

where the weight vector is $w = (2^n, -2^{n-1}, \ldots, -1)$ and the threshold is 0. We will not discuss lower bounds for LTFs, because they are ludicrously simple: there's no halfspace which can properly "classify" $\textsc{parity}(x_1, x_2)$, so $\textsc{parity}(x_1, x_2)$ is not computed by an LTF. PRGs for LTFs are not quite so simple; we will focus on these and deterministic approximate counting for LTFs.

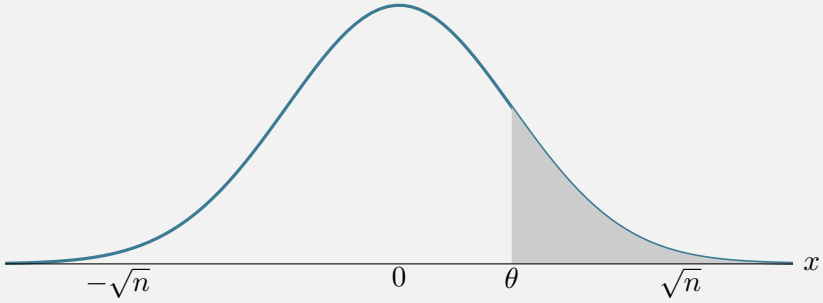
Question: Can we, in $\operatorname{poly}(n)$ time, given an LTF (w, θ), output the number of satisfying assignments $|f^{-1}(1)|$ *exactly*?
$\triangleright$ No: this is #P-hard, and so we must settle for approximate counting.

For the LTF $\operatorname{sign}(w \cdot x - \theta)$, there are 2^n values of $w \cdot x - \theta$ as x ranges over $\{\pm 1\}^n$, and this indeed forms a (really, various) discrete distribution(s) over $\mathbb{R}$. The reason is that the distribution is uniform over the 2^n values of $w \cdot x - \theta$ as x ranges over $\{\pm 1\}^n$. We can think of this as a discrete distribution over $\mathbb{R}$ with 2^n point masses. This perspective on the class of linear threshold functions is useful because later on when we discuss PRGs for LTFs, we will be able to use ideas of randomness to discuss things like regularity, and so on.

Example 10.14. 1. For $w \cdot x = x_1$, the distribution is $\mathcal{U}_{\pm 1}$.

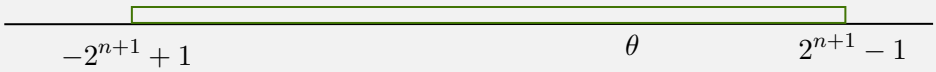


2. For $w \cdot x = x_1 + \ldots + x_n$, the distribution is $\mathsf{Bin}(n, 0) \approx \mathcal{N}(0, n)$, where $\textsc{Bin}(n, 0)$ is the binomial distribution with n trials and success probability 0.



The shaded region gives the number of satisfying assignments.

3. For $w \cdot x = \sum_{i=1}^{n} 2^i \cdot x_i$, the distribution is $\mathcal{U}_{\pm 2^n}$.



4. While the above three may seem obvious, there are arbitrarily annoying LTFs out there, for example, $w \cdot x = \sum_{j=1}^{n} j^{\log^2 j} \cdot x_j$.

The main idea of this unit is that the best LTF of all of these is the blue one in the second example—it is the Gaussian distribution, and the law of large numbers tells us that if we add a bunch of independent random variables, we get a Gaussian distribution (modulo some scaling). We will see more about this in particular and LTFs more generally in the next lecture.

§11 Lecture 11—02nd April, 2024

Last time. In the previous class, we:

- Finish the proof of Linial-Mansour-Nisan's theorem (the result about Fourier weight under restrictions).
- Proof of Braverman's theorem:
  - Augumenting Beigel-Reingold-Spielman's result: error function E is computable in AC^0.
  - Refined sandwiching goal: f' is close to f under a distribution combining $\mathcal{D}$ and $\mathcal{U}$; q_ℓ is a lower-sandwicher for f'.
  - Combining the two results: f is close to f' under $\mathcal{D}$.
- Start a new unit on linear threshold functions (LTFs): basics and examples.

Today. We will focus on LTFs:

- Regularity for LTFs, the Berry-Esseen theorem.
- PRGs for LTFs:
  - $\widetilde{O}(1/\varepsilon^2)$-wise independence ε-fools LTFs.
  - $\widetilde{O}(1/\varepsilon)$-wise independence fools ε-regular LTFs via low-degree univariate polynomial

approximations to $\operatorname{sign}(t)$.

– Beyond regular LTFs: a structure theorem based on *critical index*, junta approximations.

Reminder: the project progress report is due in two days!

---

§11.1 Regularity of linear threshold functions and the Berry-Esseen theorem

Recall the four examples we saw towards the end of the last lecture:

$$w \cdot \boldsymbol{x} = \begin{cases} \boldsymbol{x}_1 \\ \sum_i 2^i \cdot \boldsymbol{x}_i \\ \sum_i \boldsymbol{x}_i \\ \sum_j j^{\log^2 j} \cdot \boldsymbol{x}_j. \end{cases}$$

We saw that the distribution of $w \cdot \boldsymbol{x}$ looks quite different for each of these, because $\boldsymbol{x} \sim \{\pm 1\}^n$. But suppose $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, where each $\boldsymbol{x}_i \sim \mathcal{N}(0,1)$ independently. Then for any $w \in \mathbb{R}^n$ such that $\sum w_i^2 = 1$, we will always have, with $\boldsymbol{g} = (\boldsymbol{g}_1, \ldots, \boldsymbol{g}_n) = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \boldsymbol{x}$, that $w \cdot \boldsymbol{g} = \sum w_i \boldsymbol{g}_i \sim \mathcal{N}(0,1)$. This is because the $\boldsymbol{g}_i$ are independent, and the sum of independent Gaussian random variables is Gaussian:

$$\mathcal{N}(0, \sigma_1^2) + \mathcal{N}(0, \sigma_2^2) = \mathcal{N}(0, \sigma_1^2 + \sigma_2^2).$$

So if all our $\boldsymbol{x}_i$ are Gaussian, then the distribution of $w \cdot \boldsymbol{x}$ is always Gaussian. This in some sense tells us that the "nicest" LTF for us is that which works for our "nice" distribution, the Gaussian distribution, namely:

$$\operatorname{sign}\left(\frac{1}{\sqrt{n}}(x_1 + \ldots + x_n)\right).$$

where $\boldsymbol{x}_i \sim \{\pm 1\}^n$ and the $1/\sqrt{n}$ is a normalisation factor. (This is the *majority function.*) We will think of this LTF as having the distribution $w \cdot \boldsymbol{x}$, where $\boldsymbol{x} \sim \{\pm 1\}$ which looks like a Gaussian.

The discussion above motivates the following definition of regularity for LTFs:

Definition 11.1 (Regularity of LTFs)**.** *We say that an LTF $f = \operatorname{sign}(w \cdot x - \theta)$ is* ε-regular *if:*

- *$w \in \mathbb{R}^n$ is such that $\sum w_i^2 = 1$, and*
- *each $|w_i| \leqslant \varepsilon$ for all $i \in [n]$.*

> **Example 11.2.** The majority function is ε-regular for $\varepsilon = 1/\sqrt{n}$. This is the best regularity of any LTF we will see; it is as regular as can be.

Intuitively, the notion of regularity is nice because it means that $w \cdot \boldsymbol{x} \sim \mathcal{N}(0,1)$, due to the Berry-Esseen theorem. This theorem is a quantitative version of the well-known central limit theorem (CLT), which we remind ourselves of below.

Theorem 11.1 (Central limit theorem)**.** *Let $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots$ be i.i.d. random variables with $\mathbb{E}[\boldsymbol{X}_i] = \mu$ and $\mathrm{Var}(\boldsymbol{X}_i) = \sigma^2 < \infty$. Let $\boldsymbol{S}_n = \boldsymbol{X}_1 + \ldots + \boldsymbol{X}_n$. Then*

$$\frac{\boldsymbol{S}_n - n\mu}{\sigma\sqrt{n}} \to \mathcal{N}(0,1)$$

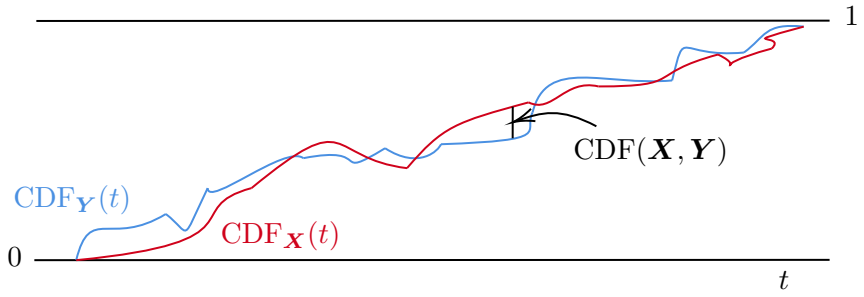
in distribution as $n \to \infty$.

Before we state the Berry-Esseen theorem, we need to define the distance between cumulative distribution functions (CDFs) of two random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$. Let

$$\mathrm{CDF}_{\boldsymbol{X}}(\theta) = \Pr[\boldsymbol{X} \leqslant \theta],$$
$$\mathrm{CDF}_{\boldsymbol{Y}}(\theta) = \Pr[\boldsymbol{Y} \leqslant \theta].$$

Then the distance between the CDFs of $\boldsymbol{X}$ and $\boldsymbol{Y}$ over some measurable space E is defined as

$$\mathrm{CDF}(\boldsymbol{X}, \boldsymbol{Y}) = \sup_{\theta \in E} \left| \mathrm{CDF}_{\boldsymbol{X}}(\theta) - \mathrm{CDF}_{\boldsymbol{Y}}(\theta) \right|.$$

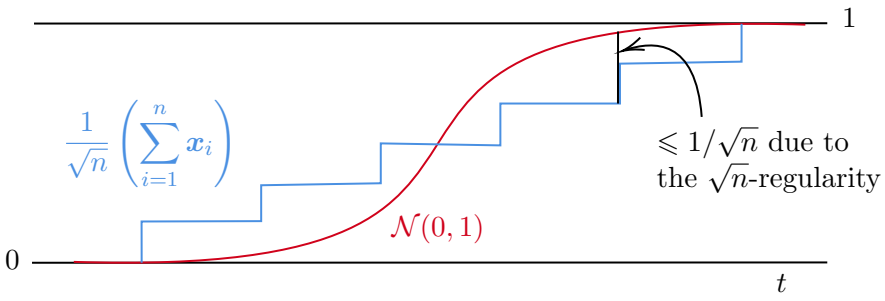


We now state the following simplified version of the theorem, which tells us something about the CDF distance between the ε-regular LTF and the Gaussian distribution.

Theorem 11.2 (Berry-Esseen theorem [Ber41, Ess42, Ess43])**.** *Let $S = \boldsymbol{X}_1 + \ldots + \boldsymbol{X}_n$ where $\boldsymbol{X}_i$ are independent real random variables with $\mathbb{E}[\boldsymbol{X}_i] = 0$ and $\mathrm{Var}[X_i] = \sum_{i=1}^n \mathbb{E}[\boldsymbol{X}_i^2] = 1$. Suppose that each $\boldsymbol{X}_i$ has $|\boldsymbol{X}_i| \leqslant \tau$ with probability 1, and let $\boldsymbol{g} \sim \mathcal{N}(0,1)$. Then for any $\theta \in \mathbb{R}$,*

$$\mathrm{CDF}(S, \boldsymbol{g}) \leqslant \tau \implies \left| \Pr_{\boldsymbol{X}}[S \leqslant \theta] - \Pr_{\boldsymbol{g} \sim \mathcal{N}(0,1)}[\boldsymbol{g} \leqslant \theta] \right| \leqslant \tau.$$

Consider the plots of two CDFs shown in red and blue in the figure below. The Berry-Esseen theorem tells us that the worst error the discrete random variable makes with respect to the standard Gaussian, as measured by the supremum of the difference in CDFs, is at most $\tau = 1/\sqrt{n}$ in this case.



Question: How does this relate to the total variation distance between the two distributions? Why not use the total variation distance instead of the CDF distance?
▷ Notice that if we were to talk about the total variation distance, then these distributions cannot look anything like each other, since the Gaussian distribution is continuous and the probability distribution function of the $\boldsymbol{X}_i = w_i \cdot \boldsymbol{x}_i$ is discrete, supported on 2^n points. Thus the total variation distance between the two distributions is always 1, regardless of the nature of the LTF. This is why we use the CDF distance, which is a more refined measure of closeness between two distributions.

At this juncture it should be clear that ε-regular LTFs are nice for our goals (*e.g.* fooling, deterministic approximate counting, counting the number of satisfying assignments, etc): suppose we are given an ε-regular LTF $f = \operatorname{sign}(w \cdot \boldsymbol{x} - \theta)$, and we want to do deterministic approximate counting, we can just output $\Pr[\boldsymbol{g} \leqslant \theta]$ where $\boldsymbol{g} \sim \mathcal{N}(0,1)$, and this will, by the Berry-Esseen theorem, be $\pm\varepsilon$-close to $\Pr[f(\boldsymbol{x}) = 1]$.

However it is not so easy to construct pseudorandom generators (PRGs) for linear threshold functions. We now focus on this problem.

§11.2 Pseudorandom generators for linear threshold functions

The main result of this section is the following theorem:

Theorem 11.3 (Diakonikolas, Gopalan, Jaiswal, Servedio, and Viola [DGJ+10])**.** *Any $\widetilde{O}(1/\varepsilon^2)$-wise independent distribution $\mathcal{D}$ over $\{\pm 1\}^n$ ε-fools the class of all LTFs.*

There are other results we will not necessarily see: it turns out that

1. the $\widetilde{O}(1/\varepsilon^2)$ is optimal up to logarithmic factors (ignoring the $\widetilde{O}$), yielding a PRG with seed length $\widetilde{O}\left(\frac{1}{\varepsilon^2} \cdot \log n\right)$.
2. if we care about the best-possible seed-length, there are hand-crafted pseudorandom generators with seed length $O\left(\log\left(\frac{1}{\varepsilon}\right) \cdot \log n\right)$.

We will not prove Theorem 11.3 above, but instead focus on the key ideas behind the proof. In particular, we will focus on the following special case of the theorem:

Theorem 11.4. *Any $\widetilde{O}(1/\varepsilon^2)$-wise independent distribution $\mathcal{D}$ over $\{\pm 1\}^n$ ε-fools the class of ε-regular LTFs.*

Recall the main way we know to show that a k-wise independent distribution fools a class of functions is by sandwiching polynomials:

Fact 11.3. *The function $f\colon \{\pm 1\}^n \to \{\pm 1\}$ is ε-fooled by any k-wise independent distribution $\mathcal{D}$ if there exist ε-sandwiching polynomials $q_\ell, q_u\colon \mathbb{R}^n \to \mathbb{R}$ such that:*

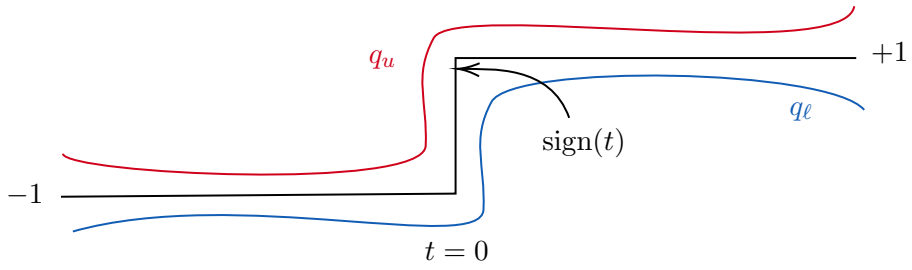
- $\deg(q_\ell), \deg(q_u) \leqslant k$,
- $q_\ell(\boldsymbol{x}) \leqslant f(\boldsymbol{x}) \leqslant q_u(\boldsymbol{x})$ *for all* $\boldsymbol{x} \in \{\pm 1\}^n$,
- $\mathbb{E}_{\boldsymbol{x} \sim \mathcal{U}_n}[q_u(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}[q_\ell(\boldsymbol{x})] \leqslant \varepsilon$.

Thus if we could show that there are sandwiching polynomials for ε-regular LTFs, we would be done. In particular, the high-level plan is to show that for any ε-regular LTF $f(x) = \mathrm{sign}(w \cdot \boldsymbol{x} - \theta)$, there exist polynomials q_ℓ, q_u of degree $O(1/\varepsilon^2)$ such that $q_\ell(\boldsymbol{x}) \leqslant f(\boldsymbol{x}) \leqslant q_u(\boldsymbol{x})$ for all $\boldsymbol{x} \in \{\pm 1\}^n$. We will argue this by giving good $\widetilde{O}(1/\varepsilon^2)$-degree approximating polynomials for univariate functions of the form $\mathrm{sign}(t)$.

In more detail, fix an ε-regular LTF $f = \mathrm{sign}(w \cdot \boldsymbol{x} - \theta)$ such that $\sum w_i^2 = 1$ and $|w_i| \leqslant \varepsilon$ for all $i \in [n]$. The Berry-Esseen theorem tells us that if $x \sim \mathcal{U}$, then the distribution $w \cdot \boldsymbol{x}$ is ε-close in CDF distance to the distribution of a single univariate Gaussian random variable $\boldsymbol{g} \sim \mathcal{N}(0,1)$. Thus, by Fact 11.3, it is enough to give univariate polynomials $q_\ell, q_u \colon \mathbb{R} \to \mathbb{R}$ of degree $\leqslant \widetilde{O}(1/\varepsilon^2)$ such that $q_\ell(t) \leqslant \mathrm{sign}(t) \leqslant q_u(t)$ for all $t \in \mathbb{R}$, and $\mathbb{E}_{\boldsymbol{g} \sim \mathcal{N}(0,1)}[q_u(\boldsymbol{g}) - q_\ell(\boldsymbol{g})] \leqslant \varepsilon$, and we can get the latter by guaranteeing that for all $t \in \mathbb{R}$,

$$\mathbb{E}_{\boldsymbol{g} \sim \mathcal{N}(0,1)}[q_u(\boldsymbol{g}) - \mathrm{sign}(t)] \leqslant \varepsilon/2, \tag{$\star$}$$

$$\mathbb{E}_{\boldsymbol{g} \sim \mathcal{N}(0,1)}[\mathrm{sign}(t) - q_\ell(\boldsymbol{g})] \leqslant \varepsilon/2. \tag{$\clubsuit$}$$

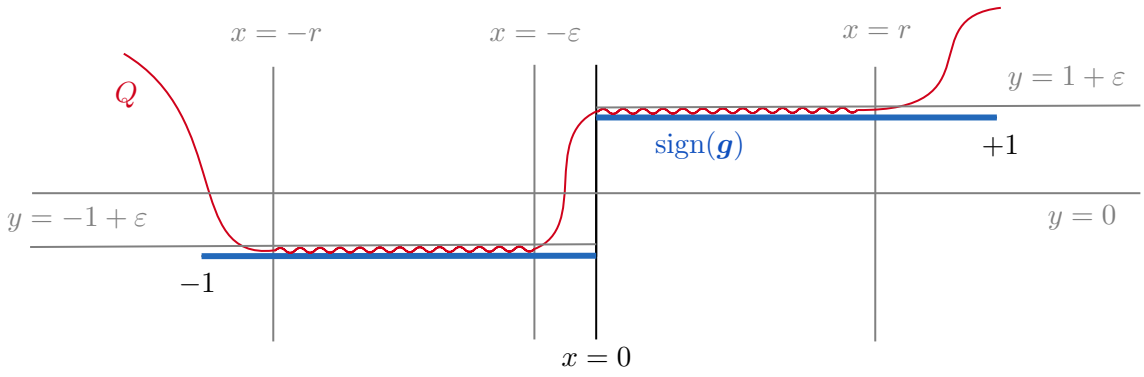


The key to getting the univariate polynomials q_ℓ, q_u is the following theorem:

Theorem 11.5. *Let $r = \widetilde{O}(1/\varepsilon^2)$. Then there is a univariate polynomial $Q \colon \mathbb{R} \to \mathbb{R}$ of degree $d \leqslant \widetilde{O}(1/\varepsilon^2)$ with the following properties:*

1. *$Q(g) \geqslant \mathrm{sign}(g) \geqslant -Q(-g)$ for all $g \in \mathbb{R}$.*
2. *$Q(g) \in [\mathrm{sign}(g), \mathrm{sign}(g) + \varepsilon]$ for all $g \in [-r, -\varepsilon] \cup [0, r]$.*
3. *$Q(g) \in [-1, 1 + \varepsilon]$ for all $g \in [-\varepsilon, 0]$.*
4. *$Q(g) \leqslant 2 \cdot (4\varepsilon g)^d$ for all $g \in \mathbb{R}$ such that $|g| \geqslant r$.*

The picture below should clarify the behaviour of the polynomial Q.



Proof sketch for why Theorem 11.5 implies (⋆) *and* (♣). Property 1 of Theorem 11.5 tells us that $Q(g) \geqslant \operatorname{sign}(g)$ and $-Q(-g) \leqslant \operatorname{sign}(g)$ for all $g \in \mathbb{R}$. So we can take $q_u(t) = Q(t)$ and $q_\ell(t) = -Q(-t)$, and we have $q_\ell(t) \leqslant \operatorname{sign}(t) \leqslant q_u(t)$ for all $t \in \mathbb{R}$. Thus we only need to verify that it is an approximator, *i.e.* that

$$\mathbb{E}_{\boldsymbol{g} \sim \mathcal{N}(0,1)}[Q(\boldsymbol{g}) - \operatorname{sign}(g)] \leqslant O(\varepsilon). \tag{⋆⋆}$$

We will argue this in three regimes: one where $|g|$ is "moderate" (not too close to 0 or 1), one where $|g|$ is "tiny" (close to 0), and one where $|g|$ is "huge" (close to 1).

Property 2 of Theorem 11.5 tells us that for these "moderate" values of g (in the majority of cases), *i.e.* those with $g \in [-r, -\varepsilon] \cup [0, r]$ pointwise, we have $|Q(g) - \operatorname{sign}(g)| \leqslant \varepsilon$. Thus the contribution to the expectation in (⋆⋆) from these values is $\leqslant O(\varepsilon)$.

Property 3 of Theorem 11.5 tells us that for the "tiny" values of g, *i.e.* those with $g \in [-\varepsilon, 0]$, we have pointwise error as large as $O(1)$, but no larger. Indeed, for $\boldsymbol{g} \sim \mathcal{N}(0,1)$, we have $\Pr[\boldsymbol{g} \in [-\varepsilon, 0]] \leqslant O(\varepsilon)$, so the contribution to the expectation in (⋆⋆) from these values is $\leqslant O(1) \cdot O(\varepsilon) = O(\varepsilon)$.

For the "huge" values of g, *i.e.* those with $|g| \geqslant r$ for which the pointwise error $|Q(g) - \operatorname{sign}(g)|$ might be arbitrarily large, property 4 of Theorem 11.5 gives us assurances that it is under control, *i.e.* $Q(g) \leqslant 2 \cdot (4\varepsilon g)^d$ for all $g \in \mathbb{R}$ such that $|g| \geqslant r$, so that the contribution to the expectation in (⋆⋆) from these values is $\leqslant 1 + 2 \cdot (4\varepsilon g)^d = \xi_1$. In addition, Gaussian tail bounds are very strong:

$$\Pr[|\boldsymbol{g}| \geqslant t] \leqslant e^{-t^2/2} = \xi_2 \quad \text{for all } t \geqslant 0.$$

So essentially we're trading off the fact that the error could be big (ξ_1) for the fact that the values of g for which the error is big (ξ_2) are very rare. The structure of the Gaussian distribution is such that the error ξ_1 wins over ξ_2, and we get that the contribution to the expectation in (⋆⋆) is $\leqslant O(\varepsilon)$. (Here's a sketch of why this is true: consider outcomes of $g \in [r, r+1]$; then the probability $\Pr[\boldsymbol{g} \in [r, r+1]] \leqslant \Pr[\boldsymbol{g} > r] \leqslant e^{-r^2/2}$. On the other hand, for such g, the error of Q is, by property 4,

$$\lesssim 2 \cdot (4\varepsilon \cdot (r+1))^d \approx \left(\operatorname{polylog}\left(\frac{1}{\varepsilon}\right)\right)^d \approx 2^{\widetilde{O}(1/\varepsilon^2)}.$$

Now by the suitable choice of the hidden $\log \frac{1}{\varepsilon}$ factors in r, we get that $e^{-r^2/2} \cdot 2^{\widetilde{O}(1/\varepsilon^2)} \ll \varepsilon/2$. A similar argument gives us that $[r+t, r+t+1]$ for $t \in \mathbb{N}$ also contributes $\ll \varepsilon/2^t$ to the expectation in (⋆⋆), and summing over all $t \in \mathbb{N}$ gives us that the total contribution from huge $g \in [r, \infty)$ is $\leqslant \sum_{t=1}^{\infty} \varepsilon/2^t = O(\varepsilon)$.)

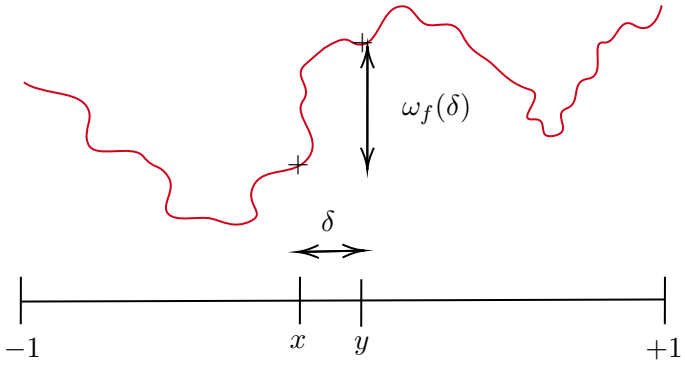
The argument for (♣) is similar, and we are done. □

We still haven't proved Theorem 11.5, nor will we, but we will prove something like Property 2 in the theorem the interesting (and majority) step using ideas from approximation theory.

Definition 11.4 (Modulus of continuity)**.** *Let $f\colon [-1,1] \to \mathbb{R}$ be bounded and continuous be defined over the compact set $[-1,1]$. The* modulus of continuity *of f is the function $\omega_f\colon [0,1] \to \mathbb{R}$ defined by*

$$\omega_f(\delta) = \sup_{|x-y| \leqslant \delta} |f(x) - f(y)|.$$

The modulus of continuity is a measure of how much f can change over a small interval. For example, if f is Lipschitz continuous with constant L, then $\omega_f(\delta) \leqslant L \cdot \delta$ for all $\delta \in [0,1]$.



A fact from approximation theory is that the modulus of continuity of a function controls the error of the best pointwise approximator to the function of a certain degree. In particular, we have the following theorem, which we will not prove.

Theorem 11.6 (Jackson's theorem [Jac12])**.** *Let $f\colon [-1,1] \to \mathbb{R}$ be bounded and continuous, and let $\ell \in \mathbb{N}$ be a positive integer with $\ell \geqslant 1$. Then there exists a polynomial $J(t)$ of degree at most ℓ such that*

$$\max_{t\in[-1,1]} |J(t) - f(t)| \leqslant 6 \cdot \omega_f\left(\frac{1}{\ell}\right).$$

We will use this theorem to prove the following lemma, which is a weaker version of Property 2 of Theorem 11.5.

Lemma 11.5. *Let $a = \widetilde{O}(\varepsilon^2)$, and let $m = \frac{1}{a} \cdot 300 \ln\left(\frac{1}{\varepsilon}\right) = \widetilde{O}(1/\varepsilon^2)$. Then there exists a polynomial $q(t)$ of degree at most m such that*

$$\max_{t\in[-1,-a]\cup[a,1]} |q(t) - \operatorname{sign}(t)| \leqslant \varepsilon.$$

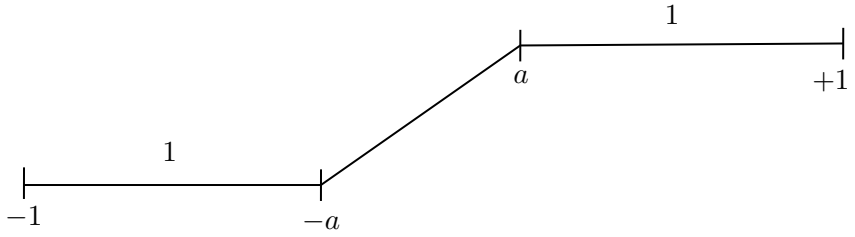
Think of $Q(g)$ in Theorem 11.5 as $Q(g) = q(g/r)$, *i.e.* $q(t) = Q(rt)$, and think of $r = \widetilde{O}(\varepsilon)$ as a small constant. Then the lemma tells us that we can rescale the arguments as

$$\begin{aligned} g(1) &= Q(r), \\ q(-a) &= Q(-ra) \approx Q(-\varepsilon), \end{aligned}$$

and so we get something like Property 2 of Theorem 11.5.

Proof of Lemma 11.5. Define the continuous function $f\colon [-1,1] \to [-1,1]$ by

$$f(x) = \begin{cases} \operatorname{sign}(x) & \text{if } |x| \in [a,1] \\ x/a & \text{if } |x| \in [0,a]. \end{cases}$$



Clearly the modulus of continuity of this function f is $\omega_f(\frac{1}{\ell}) = \frac{1}{a\ell}$, and so by Theorem 11.6 with $\ell = \frac{25}{a}$, there exists a polynomial $J(t)$ of degree ℓ such that

$$\max_{a \leqslant |t| \leqslant 1} |J(t) - \operatorname{sign}(t)| \leqslant \max_{|t| \leqslant 1} |J(t) - f(t)| \leqslant 6 \cdot \omega_f\left(\frac{1}{\ell}\right) = \frac{6}{a\ell} \leqslant \frac{1}{4},$$

where the last inequality follows from the choice of ℓ. But we want error ε instead of $\frac{1}{4}$. Now we use Jackson's theorem with an even larger degree ℓ, but in that case we would need degree about $\widetilde{\Theta}(1/\varepsilon^3)$, which is too large. So instead, we will use a low degree (degree k) *amplifying polynomial*:

$$A_k(u) = \sum_{j \geqslant k/2}^{k} \binom{k}{j} \cdot \left(\frac{1+u}{2}\right)^j \cdot \left(\frac{1-u}{2}\right)^{k-j}.$$

Note that this polynomial exactly captures the probability that the binomial distribution with a success probability of $\frac{1+u}{2}$ has at least $k/2$ successes:

$$A_k(u) = \Pr\left[\text{toss a coin with heads probability } \frac{1+u}{2} \text{ } k \text{ times, get at least } k/2 \text{ heads}\right].$$

Observe that if $u \gg 0$, then $A_k(u) \approx 1$, and if $u \ll 0$, then $A_k(u) \approx 0$—this is the amplification property of the polynomial. A direct application of the Chernoff bound tells us that:

- if $u \in [\frac{3}{5}, 1]$, then $2A_k(u) - 1 \in [1 - 2e^{-k/6}, 1]$,
- if $u \in [-1, -\frac{3}{5}]$, then $2A_k(u) - 1 \in [-1, -1 + 2e^{-k/6}]$.

Thus our final polynomial is

$$q(t) = 2A_k\left(\frac{4}{5}J(t)\right) - 1.$$

(Here we are scaling $J(t)$ by $\frac{4}{5}$ to ensure that $\frac{4}{5}J(t) \in [-1, -\frac{3}{5}] \cup [\frac{3}{5}, 1]$.) So $k = 12 \log \frac{1}{\varepsilon}$, we have that $2e^{-k/6} \leqslant \frac{\varepsilon}{2} < \varepsilon$, and everything works out. Finally we check the degree:

$$\deg(q) \leqslant \deg(J) \cdot \deg(A_k) \leqslant \frac{25}{a} \cdot 12 \log \frac{1}{\varepsilon} = \frac{300 \log \frac{1}{\varepsilon}}{a^2} = m,$$

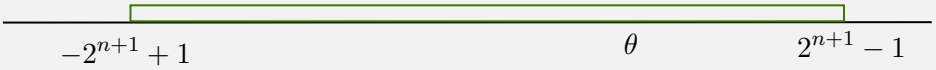
and so we're getting the right approximation of $\operatorname{sign}(t)$ for $t \in [-1, -a] \cup [a, 1]$. □

So we can claim a moral victory: we have victory *vis-à-vis* fooling ε-regular linear threshold functions. However, not every linear threshold function is ε-regular.

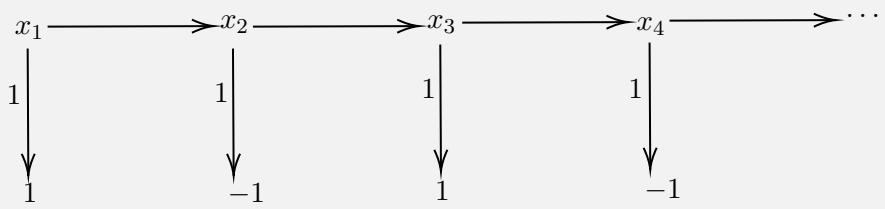
Example 11.6. The linear threshold function

$$f(x) = \operatorname{sign}(2^n x_1 - 2^{n-1} x_2 - 2^{n-2} x_3 - \cdots - 2x_n - \theta)$$

is only $\Theta(1)$-regular, but yet we need to fool LTFs like this. The actual plot of $w \cdot x$ for this w is uniform:



So we shouldn't have much hope that our strategy above should work for LTFs like this. However, this LTF is not particularly hard; it is the following decision list:



and this decision list is ε-close to a $\log \frac{1}{\varepsilon}$-junta.

Now, is every non-regular LTF close to a junta? Unfortunately, the answer is no. But we can still make some progress. Suppose our LTF is $f(x) = \text{sign}(w \cdot x - \theta)$, where $w \in \mathbb{R}^n$ is such that $\sum_{i=1}^n w_i^2 = 1$. and f is not ε-regular. Say, without loss of generality, that $|w_1| \geqslant |w_2| \geqslant \cdots \geqslant |w_n|$. The fact of f's non-regularity means that $|w_1| \geqslant \varepsilon$. Now, take x_1 and set it aside, so that we now know that the sum $w_2^2 + \cdots + w_n^2 = 1 - w_1^2$ goes down non-trivially to $\leqslant 1 - \varepsilon^2$. This linear form over $(w_2, \ldots, w_n)$ might then be regular (the non-regularity of f could be "concentrated" in w_1), which we already know how to handle. If not, we can repeat the process with w_2, and so on. This action of "chasing regularity" is the idea behind the following definition:

Definition 11.7 (Critical index of a linear threshold function)**.** *Fix $f(x) = \text{sign}(w \cdot x - \theta)$, where $w \in \mathbb{R}^n$ is such that $\sum_{i=1}^n w_i^2 = 1$, to be a not-necessarily-ε-regular LTF, and suppose without loss of generality that $|w_1| \geqslant |w_2| \geqslant \cdots \geqslant |w_n|$. The* ε-critical index *of f is the minimum value ℓ such that the linear form $(w_\ell, w_{\ell+1}, \ldots, w_n)$ is ε-regular, i.e.*

$$|w_\ell| \leqslant \varepsilon \sqrt{\sum_{j=\ell}^n w_j^2}.$$

This definition immediately gives the following fact:

Fact 11.8. *If $\ell(\varepsilon)$ is the ε-critical index of $(w_1, \ldots, w_n)$, where $\sum_{i=1}^n w_i^2 = 1$, then*

$$\sum_{j=\ell(\varepsilon)}^n w_j^2 \leqslant (1 - \varepsilon^2)^{\ell(\varepsilon)-1}.$$

This fact is nice, because it puts us in a win-win situation: for a general halfspace, we can readily ask for what the critical index is. In particular, given any $f = \text{sign}(w \cdot x - \theta)$, we can consider three cases based on the ε-critical index $\ell(\varepsilon)$ of f:

1. *Case 1:* $\ell(\varepsilon) = 1$. In this case, f is ε-regular, and we are done.
2. *Case 2:* $\ell(\varepsilon) \leqslant K/\varepsilon^2$ *for some constant K.* We can think of this case as one where $w \cdot x$ has a "junta part" $w_1 x_1 + \cdots + w_{\ell(\varepsilon)} x_{\ell(\varepsilon)}$, and a "regular part" $w_{\ell(\varepsilon)+1} x_{\ell(\varepsilon)+1} + \cdots + w_n x_n$. We can then apply a junta theorem to the junta part, and the regularity theorem to the regular part, and we'd be done.
3. *Case 3:* $\ell(\varepsilon) > K/\varepsilon^2$ *for some constant K.* By Fact 11.8, we have that

$$\sum_{j=\ell(\varepsilon)}^n w_j^2 \leqslant (1 - \varepsilon^2)^{\ell(\varepsilon)-1} \leqslant e^{-\varepsilon^2 \ell(\varepsilon)} \leqslant e^{-K}.$$

Take $k = 100 \ln \frac{1}{\varepsilon}$, so that $e^{-K} \leqslant \varepsilon^{100}$; the tail is quite tiny. Indeed, we can show in this case that f is very close to a (K/ε^2)-junta.

Along the lines of our discussion above, we can prove a "structure theorem" for linear threshold functions:

Theorem 11.7 (Structure theorem for LTFs)**.** *Fix any $\varepsilon > 0$, and any linear threshold function $f(x) = \mathrm{sign}(w \cdot x - \theta)$. Then there is a set $H \subseteq [n]$ of $\widetilde{O}(1/\varepsilon^2)$-many variables of f (the ones with the largest $|w_i|$s) such that either:*

(a) $f_{\restriction \rho}$ is ε-regular for every restriction ρ fixing the variables in H (like cases 1 and 2 above), or

(b) f is ε^{100}-close to a $|H|$-junta.

This theorem can be used to prove Theorem 11.3 (see [DGJ+10] for more details):

(a) $\widetilde{O}(1/\varepsilon^2)$-wise independence handles H and another $\widetilde{O}(1/\varepsilon^2)$-wise independent distribution handles every ε-regular restricted $f_{\restriction \rho}$, and

(b) $\widetilde{O}(1/\varepsilon^2)$-independence fools any $|H|$-junta.

Next time we will say things about fooling polynomial threshold functions.

§12 Lecture 12—09th April, 2024

Last time. In the previous class, we talked about linear threshold functions:

- Regularity, the Berry-Esseen theorem.
- Pseudorandom generators for LTFs: $\widetilde{O}(1/\varepsilon^2)$-wise independence ε-fools LTFs.
  - $\widetilde{O}(1/\varepsilon)$-wise independencefools ε-regular LTFs via low-degree univariate polynomial approximations to $\mathrm{sign}(t)$.
  - Beyond regular LTFs: a structure theorem based on *critical index*, junta approximations.

Today. We will discuss a quick high-level sketch of:

- Deterministic approximate counting for LTFs (relative error) and deterministic approximate counting for PTFs (absolute error).
- The Nisan-Widgerson PRG: a generic way to get PRGs for $\mathcal{C}$ from average-case lower bounds for a related class $\mathcal{C}'$.

Reminder: this is the last lecture for the semester. Project presentations will be held next week and the week after.

§12.1 Sketch of deterministic approximate counting for LTFs and PTFs

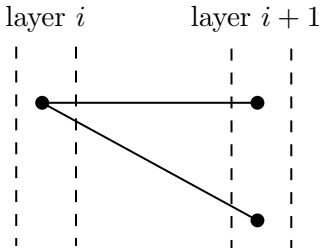
The problem of deterministic approximate counting for LTFs is this: given inputs $w_1, \ldots, w_n, \theta$ and $\varepsilon > 0$, we want to output an approximation $\hat{N}$ such that $N \leqslant \hat{N} \leqslant (1+\varepsilon)N$ where $N = \sum_{i=1}^{n} \mathbb{1}\{w_i \cdot x \geqslant \theta\}$. is the true number of satisfying assignments of the LTF $f(x) = \mathrm{sign}(w \cdot x - \theta)$. The first randomised algorithm for this problem was presented at the turn of the century [MS04] and was fairly complicated; a few years later a simpler algorithm based on dynamic programming was presented [Dye03]. Today we will give a sketch of a deterministic algorithm for this problem. Let $w_i, \theta \in \mathbb{Z}$ and let $W = \max\{|w_1|, \ldots, |w_n|, |\theta|\}$.

Theorem 12.1. *There is a* $\mathrm{poly}\left(n, \log W, \frac{1}{\varepsilon}\right)$*-time deterministic algorithm that outputs an approximation* $\hat{N}$ *such that* $N \leqslant \hat{N} \leqslant (1+\varepsilon)N$.

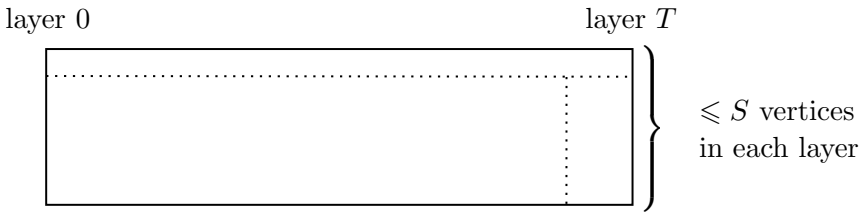
The idea here is to:

- construct a suitable (read-once) branching program which serves as a high-accuracy approximation to the LTF.
- use a standard dynamic programming method to exactly count the number of satisfying assignments of the branching program.

Definition 12.1 (Branching program)**.** *An* (S, T)-branching program *is a layered digraph with* $0, 1, \ldots, T$ *layers, where each layer* i *has at most* S *states (vertices). The first layer has a single vertex* v_0 *(the start state) and the last layer is labelled with either* 0 *(reject) or* 1 *(accept). For* $0 \leqslant i \leqslant T$*, a vertex* v *in layer* i *has two outgoing edges labelled* $0, 1$ *and ending at vertices in layer* $i+1$.



Clearly a branching program computes a function $f\colon \{0,1\}^T \to \{0,1\}$ in the natural way: start at v_0 and follow the edges according to the input bits. The function computed by a branching program is the characteristic function of the set of paths from v_0 to an accept state.

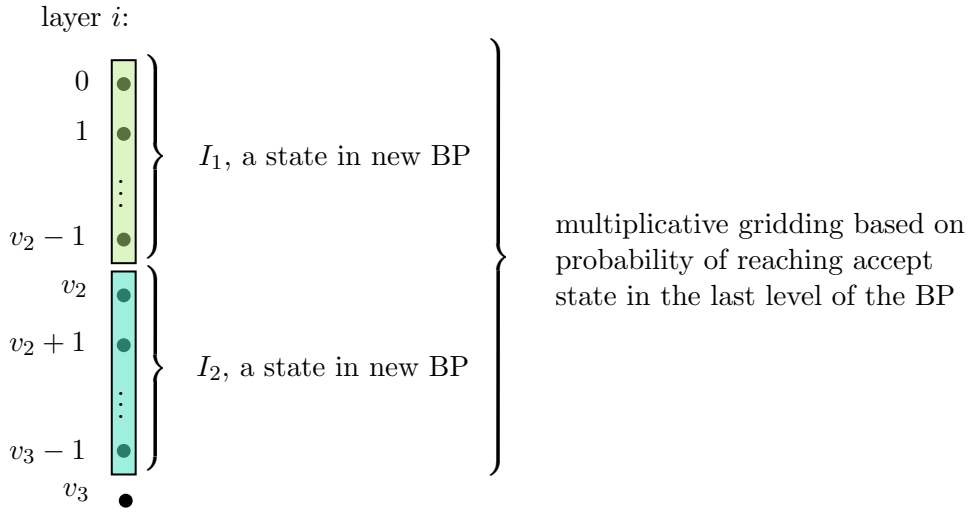


What is the connection to LTFs? Without loss of generality suppose $w_i, \theta \geqslant 0$, and let $W = \sum_{i=1}^{n} w_i$. Then we can see that an LTF f is computed by a $(W+1, n)$-branching program: each state in layer j corresponds to a possible integer value of $w_1x_1 + \ldots + w_jx_j$.

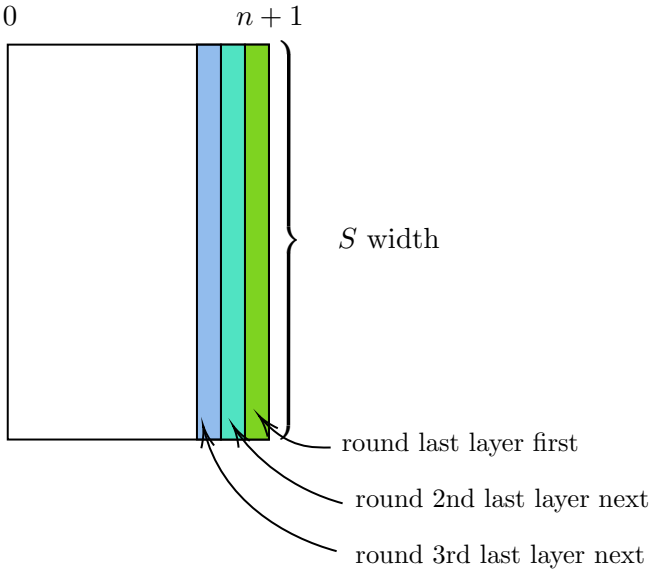
It turns out that by keeping track of the number of ways to reach each state in the branching program, we can compute the number of satisfying assignments of the branching program efficiently:

Fact 12.2. *We can exactly compute the number of satisfying assignments of an (S,T)-branching program in time* $\mathrm{poly}(S,T)$ *via dynamic programming.*

To get this down from time $\mathrm{poly}(W)$ to $\log(W)$, we then argue that given an LTF $f(x) = \mathrm{sign}(w \cdot x - \theta)$, we can construct a branching program with enough states. In particular, we can approximate the $(W+1, n)$-branching program for f with a $(\mathrm{poly}\log W, n+1)$-branching program such that the states of the approximating branching program correspond to subsets of states of the original branching program.



(This is the process of *rounding* the branching program.) Note that the probability of reaching an accept state in the last layer of the branching program is exactly the fraction of satisfying assignments of the LTF, which is what we want to compute in the first place (we need to solve the problem we're trying to curate a method for). The idea is to use the dynamic programming algorithm to do n successive stages of rounding in a "back to front" manner, where at each stage we round the branching program to a smaller branching program with fewer states.



§12.2 A brief segue into polynomial threshold functions

Definition 12.3 (Polynomial threshold functions)**.** *A* polynomial threshold function (PTF) *is a function* $f\colon \mathbb{R}^n \to \{0,1\}$ *of the form* $f(x) = \operatorname{sign}(p(x))$ *where* $p(x) = \sum_{S\subseteq[n]} a_S \prod_{i\in S} x_i$ *is a degree-*d *real polynomial.*

Here are some things we know about PRGs for PTFs:

- [MZ10] gives a pseudorandom generator for PTFs with seed length $\left(\frac{d}{\varepsilon}\right)^{O(d)} \cdot \log n$ by using a "monotone trick" to show that PRGs for poly-width branching programs can be used to fool linear threshold functions, and then extending this PRG for linear threshold functions to high-degree polynomial threshold functions. With this seed length, we can do approximate counting for PTFs in time $n^{\left(\frac{d}{\varepsilon}\right)^{O(d)}}$.
- [Kan17] gives a pseudorandom generator for PTFs with seed length $O_d(1) \cdot \operatorname{poly}(\frac{1}{\varepsilon}) \cdot \log n$ via a structure theorem for poorly anticoncentrated polynomials of Gaussians. With this seed length, we can do approximate counting for PTFs in time $n^{O_d(1)\cdot \operatorname{poly}\frac{1}{\varepsilon}}$.

We don't know how to do better than this in the sense of getting a PRG for PTFs with smaller seed length. We saw, however, that we can do better for approximate counting for LTFs than with PRGs for LTFs. It turns out that we can also do better with approximate counting for PTFs than with PRGs for PTFs; in particular, we can do $O_{d,\varepsilon}(1) \cdot n^{O(d)}$-time deterministic approximate counting for degree-d PTFs with relative error ε.

We will not go into a lot of detail for how this is done, but we will sketch some of the ingredients for the case $d = 2$ as follows. A nice extension of the Berry-Esseen theorem (Theorem 11.2)—which we can think of as saying that a regular linear form in independent random variables "behaves like" a Gaussian—is the *invariance principle* due to Mossel, O'Donnell, and Oleszkiewicz [MOO05]. This principle is a powerful generalisation of Berry-Esseen that says that once we define a notion of regularity for low-degree polynomials, then a regular low-degree polynomial in independent random variables "behaves like" a polynomial in Gaussian random variables. Then we can compare the distribution of $p(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$ where $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ are independent random variables from $\{-1, 1\}$ to the distribution of $p(\boldsymbol{g}_1, \dots, \boldsymbol{g}_n)$ where $\boldsymbol{g}_1, \dots, \boldsymbol{g}_n$ are independent standard Gaussians. This result then opens the door to using geometric arguments for high-dimensional Gaussians as well as other powerful theory to analyse the distribution of polynomials with standard Gaussian inputs.

§12.3 Hardness versus randomness: the Nisan-Widgerson pseudorandom generator

The Nisan-Widgerson pseudorandom generator [NW94] is a generic PRG giving a PRG for a class $\mathcal{C}$ of n-variable Boolean functions from an average-case lower bound against a "richer" class $\mathcal{C}'$ of r-variable Boolean functions.

The Nisan-Widgerson generator $G(\boldsymbol{u}_1, \dots, \boldsymbol{u}_s)$ is an n-bit pseudorandom string, where the truly random input bits $\boldsymbol{u}_1, \dots, \boldsymbol{u}_s$ are drawn from $\{0,1\}^s$. The generator works like this. Let S be a special sequence of n subsets such that $S = (S_1, \dots, S_n)$ where each $S_i \subseteq [s]$ and $|S_1| = \dots = |S_n| = r$. Now let $h \colon \{0,1\}^r \to \{0,1\}$ be an average-case hard function for the class $\mathcal{C}'$. Then the generator G is as follows:

$$G(\underbrace{\boldsymbol{u}_1, \dots, \boldsymbol{u}_s}_{=\mathcal{U} \in \{0,1\}^s}) = \overbrace{(h(\underbrace{\mathcal{U}|_{S_1}}_{r \text{ bits}}), h(\underbrace{\mathcal{U}|_{S_2}}_{r \text{ bits}}), \dots, h(\underbrace{\mathcal{U}|_{S_n}}_{r \text{ bits}}))}^{n \text{ bits}}, \tag{$\star$}$$

where $\mathcal{U}|_{S_i}$ denotes the r-bit string obtained by taking the bits of $\mathcal{U}$ indexed by S_i, and the sets $S_1, \dots, S_n \subset [s]$ are "almost-disjoint" from each other, *i.e.* each pair S_i, S_j has $|S_i \cap S_j|$ very small. The high-level intuition for the construction of this function is that the hardness of the function h should prevent an adversary from precisely predicting each bit of the output of the generator—we will make this intuition more precise in a moment. Before that, the "richer" class $\mathcal{C}'$ of functions from $\{0,1\}^r$ to $\{0,1\}$ is defined as $C' \coloneqq \mathcal{C} \circ \mathsf{Junta}_{r,k}$, *i.e.* for $f \colon \{0,1\}^n \to \{0,1\}$ and $g_i \colon \{0,1\}^r \to \{0,1\}$,

$$\mathcal{C}' = \{f(g_1(x), \dots, g_n(x)) : f \in \mathcal{C} \text{ and each } g_i \text{ is a } k\text{-junta}\}.$$

We have the following theorem due to Nisan and Widgerson:

Theorem 12.2 (Nisan-Widgerson [NW94])**.** *Let $\mathcal{C}$ be a class of functions from $\{0,1\}^n$ to $\{0,1\}$. Let $h \colon \{0,1\}^r \to \{0,1\}$ be an ε-hard function for $\mathcal{C}' = \mathcal{C} \circ \mathsf{Junta}_{r,k}$. Let $S = (S_1, \dots, S_n)$ be an (s, r, k)-combinatorial design. Then the generator $(\star)$ defined above (εn)-fools $\mathcal{C}$ and has seed length s.*

Before we prove this theorem, we need to define and prove the existence of these combinatorial designs. But even before that, here are a few remarks about this theorem:

- We need $\varepsilon \ll \frac{1}{n}$ for the generator to be useful. This is because the generator is only (εn)-fooling the class $\mathcal{C}$. If ε is too large (on the order of n^{-1}), then the generator is not very useful.

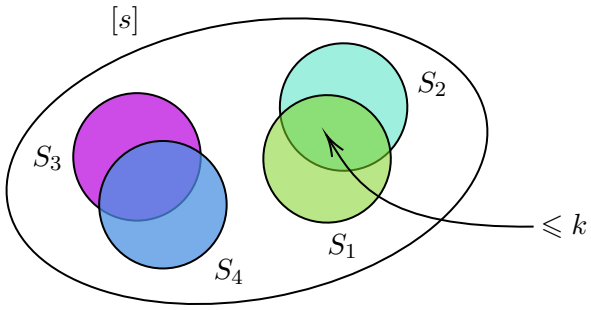
- We also need $s \ll n$ for the generator to be useful. This is because the generator has seed length s, and we want the seed length to be much smaller than the input length; otherwise there's no point constructing the pseudorandom generator.
- As we will see, the definition of the combinatorial design requires that $r \leqslant s$.

We only need to define and prove the existence of combinatorial designs, restate the theorem above more specifically, and then prove it.

Combinatorial designs A combinatorial design is a collection of subsets of $[s]$ with certain properties. We define it as follows:

Definition 12.4 (Combinatorial design)**.** *Let $s, r, k \in \mathbb{N}$. A collection S of subsets $S_1, \ldots, S_n \subseteq [s]$ is an (s, r, k)-combinatorial design if the following properties hold:*

1. *Each S_i has size r, i.e. $|S_i| = r$ for all i.*
2. *Each pair S_i, S_j has intersection of size at most k, i.e. $|S_i \cap S_j| \leqslant k$ for all $i \neq j$.*



The dream is to have $s \ll n$ (since s is the seed length of the generator), and k very small (otherwise $\mathsf{Junta}_{r,k}$ is more powerful and by extension $\mathcal{C}'$ is more powerful, which would then make it difficult for h to be hard for $\mathcal{C}'$). Thus in some sense we have a tension among parameters, and one sense in which there is this tension is that a smaller k implies a larger s (since we need more bits to index the sets S_i and account for their overlaps) while we definitely need to have $s \geqslant \log n$ (if $s < \log n$, then no matter how big the sets S_i are, we can't have n distinct subsets).

We need to construct this design, because the PRG evaluates the hard function on all of the different sets in this combinatorial design; otherwise we have no hope of obtaining an explicit PRG. Fortunately we can do this and it is not particularly hard:

Lemma 12.5. *Let $c \geqslant 1$. Furthermore, let $s = 100c^2 \log n$, $r = c \cdot \log n$, and $k = \log n$. Then there exists a greedy algorithm to construct $S_1, \ldots, S_n$ satisfying the properties of an (s, r, k)-combinatorial design, and this algorithm runs in $\leqslant \mathrm{poly}(n) \cdot 2^s$ time.*

Proof. The greedy algorithm acts as follows: across all n stages, it tries all $|S_i| = r$ at stage i, looking for one such that $|S_i \cap S_j| \leqslant k$ for all $j < i$. If it finds such an S_i, it adds it to the design; otherwise, it fails. The algorithm runtime is as claimed, because at each stage i, there are $\binom{s}{r} \leqslant 2^s$ choices for S_i, and since there are n stages, the total runtime is $\leqslant \mathrm{poly}(n) \cdot 2^s$. Thus it remains only to prove correctness.

We will show that after picking the sets $S_1, \ldots, S_{i-1}$ and fixing these, a *random* $\boldsymbol{S} \subset [s]$ satisfying $|\boldsymbol{S}| = r$ has

$$\Pr_{\boldsymbol{S}}[|\boldsymbol{S} \cap S_j| > k \text{ for some } j \in [i-1]] < 1.$$

If this holds, then we are done, because then there must exist some S_i that satisfies the second property of the combinatorial design. In fact, this follows from showing that for $r = c \cdot \log n$,

$$\Pr_{\boldsymbol{S}}[|\boldsymbol{S} \cap \{1, \ldots, c \log n\}| > k] < \frac{1}{n}.$$

Towards this, note that since $k = \log n$, we have

$$\mathbb{E}_{\boldsymbol{S}}[|\boldsymbol{S} \cap \{1, \ldots, c \log n\}|] = c \log n \cdot \frac{c \log n}{100c^2 \log n} = \frac{\log n}{100}.$$

Thus by a Chernoff bound over all the negatively correlated random variables, we get that

$$\Pr_{\boldsymbol{S}}[|\boldsymbol{S} \cap \{1, \ldots, c \log n\}| \geqslant \log n] \leqslant \exp\left(-\frac{c}{3} \cdot \log n\right) = \frac{1}{n^{\frac{c}{3}}}.$$

Summing over all n stages, we get that the probability of failure is at most $n \cdot \frac{1}{n^{\frac{c}{3}}} = \frac{1}{n^{\frac{c}{3}-1}} < 1$ for $c > 3$. Thus the algorithm is good enough. □

Proof of Nisan-Wigderson pseudorandom generator theorem Given the above construction, we now state a more precise version of the theorem above:

Theorem 12.3. *Let $\mathcal{C}$ be a class of functions from $\{0,1\}^n$ to $\{0,1\}$. Let $h\colon \{0,1\}^r \to \{0,1\}$ be an ε-hard function for $\mathcal{C}' = \mathcal{C} \circ \mathsf{Junta}_{r,k}$. Let $S = (S_1, \ldots, S_n)$ be an (s, r, k)-combinatorial design with $s = 100c^2 \log n$, $r = c \cdot \log n$, and $k = \log n$. Then there exists a generator G with seed length s that (εn)-fools $\mathcal{C}$. The generator G is defined as in ($\star$).*

Before we prove this theorem, we first present a brief discussion of how it may be applied to fool AC^0 circuits.

Corollary 12.6. *The Nisan-Widgerson generator ($\star$) gives a δ-PRG for $\mathrm{AC}^0_{M,d}$ (for $M \geqslant n$) with seed length $s = \left(\log\left(\frac{M}{\delta}\right)\right)^{2d+O(1)}$, computable in $\mathrm{poly}(n) \cdot 2^s$ time.*

Proof. The corollary follows by using the average-case lower bounds against AC^0 we proved earlier and the paraneter settings $r = c \cdot \log n$, $k = \log n$, and $s = 100c^2 \log n$ in the combinatorial design for the Nisan-Widgerson generator theorem, and then noting that $\mathrm{AC}^0_{M,d} \circ \mathsf{Junta}_{r,k} \subseteq \mathrm{AC}^0_{M+n\cdot 2^k, d+2}$. The runtime follows from the construction of the combinatorial design as well as some extra constant-time and poly-time work, so the run-time for the combinatorial design dominates. The seed length also follows from our discussion. □

Now we launch into the proof. The proof of Theorem 12.3 relies on the key notion of "next-bit unpredictability," which is an equally valid way of thinking of a pseudorandom distribution in the sense that it obscures every opportunity for an omniscient adversary to predict the n-th bit with any advantage over random choice given the first $n-1$ bits.

Definition 12.7 (Next-bit unpredictability)**.** *Let $\boldsymbol{X}$ be a random variable over $\{0,1\}^n$, and let $f\colon \{0,1\}^n \to \{0,1\}$ be a function. Also, let $\varepsilon > 0$. We say that $\boldsymbol{X}$* is ε-next-bit-unpredictable for f *if for each $i \in [n]$ and for each $a \in \{0,1\}^{n-i+1}$, we have*

$$\left| \Pr_{\boldsymbol{X}} \left[f\left(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{i-1}, a\right) = \boldsymbol{X}_i \right] - \frac{1}{2} \right| \leqslant \varepsilon,$$

or equivalently, $\boldsymbol{X}$ ε-fools the mapping $x \mapsto f(x_1, \ldots, x_{i-1}, a) \oplus x_i$.

The proof[8] of Theorem 12.3 will rely on lemmas that prove each of the following assertions:

1. The generator ($\star$) is ε-next-bit-unpredictable for each $f \in \mathcal{C}$.
2. If a random variable $\boldsymbol{X}$ is ε-next-bit-unpredictable for all $f \in \mathcal{C}$, then $\boldsymbol{X}$ (εn)-fools every $f \in \mathcal{C}$.

We prove these in order.

Lemma 12.8. *Under the conditions of Theorem 12.3, the generator ($\star$) is ε-next-bit-unpredictable for all $f \in \mathcal{C}$.*

Proof. Fix any $f\colon \{0,1\}^n \to \{0,1\}$, any $i \in [n]$ and any $a \in \{0,1\}^{n-i+1}$. Let $\boldsymbol{u} \sim \mathcal{U}_s$, and let $\boldsymbol{X} = G(\boldsymbol{u})$ be an n-bit string generated by the Nisan-Widgerson generator. We want to show that $\boldsymbol{X}$ is ε-next-bit-unpredictable for f. We have

$$\begin{aligned}\left| \Pr_{\boldsymbol{X}} \left[f(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{i-1}, a) = \boldsymbol{X}_i \right] - \frac{1}{2} \right| &= \left| \mathbb{E}_{\boldsymbol{u}_{[s]\setminus S_i}} \left[\Pr_{\boldsymbol{u}_{S_i}} \left[f\left(h(\boldsymbol{u}|_{S_1}), \ldots, h(\boldsymbol{u}|_{S_{i-1}}), a\right) = h(\boldsymbol{u}|_{S_i}) \right] \right] - \frac{1}{2} \right| \\ &\leqslant \mathbb{E}_{\boldsymbol{u}_{[s]\setminus S_i}} \left[\left| \Pr_{\boldsymbol{u}_{S_i}} \left[f\left(h(\boldsymbol{u}|_{S_1}), \ldots, h(\boldsymbol{u}|_{S_{i-1}}), a\right) = h(\boldsymbol{u}|_{S_i}) \right] - \frac{1}{2} \right| \right].\end{aligned}$$

Now for each fixing of $\boldsymbol{u}_{[s]\setminus S_i}$, write $\boldsymbol{z} = \boldsymbol{u}_{S_i}$. For each $j < i$, since we fixed $\boldsymbol{u}_{[s]\setminus S_i}$ and $|S_j \cap S_i| \leqslant k$, we have that there is a k-junta g_j such that $h(\boldsymbol{u}|_{S_i}) = g_j(\boldsymbol{z})$. Thus we can write

$$\begin{aligned}\left| \Pr_{\boldsymbol{u}_{S_i}} \left[f\left(h(\boldsymbol{u}|_{S_1}), \ldots, h(\boldsymbol{u}|_{S_{i-1}}), a\right) = h(\boldsymbol{u}|_{S_i}) \right] - \frac{1}{2} \right| &= \left| \Pr_{\text{unif. } \boldsymbol{z}} \left[f\left(g_1(\boldsymbol{z}), \ldots, g_{i-1}(\boldsymbol{z}), a\right) = h(\boldsymbol{z}) \right] - \frac{1}{2} \right| \\ &\leqslant \varepsilon,\end{aligned}$$

where the last inequality follows from the fact that h is ε-hard for $\mathcal{C}' = \mathcal{C} \circ \mathsf{Junta}_{r,k}$. □

Lemma 12.9. *Let $\boldsymbol{X}$ be a random variable over $\{0,1\}^n$, and let $f\colon \{0,1\}^n \to \{0,1\}$ be a function. Also, let $\varepsilon > 0$. If the random variable $\boldsymbol{X}$ is ε-next-bit-unpredictable for all $f \in \mathcal{C}$, then $\boldsymbol{X}$ (εn)-fools every $f \in \mathcal{C}$.*

Proof. The proof is by a hybrid argument. Let $\boldsymbol{B} = (\boldsymbol{B}_1, \ldots, \boldsymbol{B}_n) \sim \mathcal{U}_n$ be a truly random n-bit string. We "walk" from the truly random string $\boldsymbol{B}$ to the pseudorandom string $\boldsymbol{X}$ in n steps:

[8] Our parameter settings for the combinatorial design are achieved with convenience in mind; one can readily pick other s, r, k.

$$
\begin{aligned}
\mathcal{D}_0 &= (\boldsymbol{B}_1, \ldots, \boldsymbol{B}_n) = \boldsymbol{B}, \\
\mathcal{D}_1 &= (\boldsymbol{X}_1, \boldsymbol{B}_2, \ldots, \boldsymbol{B}_n), \\
&\vdots \\
\mathcal{D}_{i-1} &= (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{i-1}, \boldsymbol{B}_i, \ldots, \boldsymbol{B}_n), \\
\mathcal{D}_i &= (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{i-1}, \boldsymbol{X}_i, \boldsymbol{B}_{i+1}, \ldots, \boldsymbol{B}_n), \\
\mathcal{D}_{i+1} &= (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{i-1}, \boldsymbol{X}_i, \boldsymbol{X}_{i+1}, \boldsymbol{B}_{i+2}, \ldots, \boldsymbol{B}_n), \\
&\vdots \\
\mathcal{D}_n &= (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n) = \boldsymbol{X}.
\end{aligned}
$$

By the triangle inequality, we have

$$
\begin{aligned}
|\mathbb{E}[f(\boldsymbol{X})] - \mathbb{E}[f(\boldsymbol{B})]| &= |\mathbb{E}[f(\mathcal{D}_n)] - \mathbb{E}[f(\mathcal{D}_0)]| \\
&\leqslant \sum_{i=1}^{n} |\mathbb{E}[f(\mathcal{D}_i) - f(\mathcal{D}_{i-1})]|.
\end{aligned}
$$

Fix $i \in [n]$. Since $-p = (1-p) - 1$ for any $p \in [0,1]$, we can write

$$
\begin{aligned}
&|\mathbb{E}[f(\mathcal{D}_i) - f(\mathcal{D}_{i-1})]| \\
&\quad = \left| \mathbb{E}\left[f(\mathcal{D}_{i-1}) \mid \boldsymbol{B}_i = \boldsymbol{X}_i\right] - \frac{1}{2}\mathbb{E}\left[f(\mathcal{D}_{i-1}) \mid \boldsymbol{B}_i = \boldsymbol{X}_i\right] + \frac{1}{2}\mathbb{E}\left[f(\mathcal{D}_{i-1}) \mid \boldsymbol{B}_i \neq \boldsymbol{X}_i\right] \right| \\
&\quad = \left| \frac{1}{2}\mathbb{E}[f(\mathcal{D}_{i-1}) \mid \boldsymbol{B}_i = \boldsymbol{X}_i] + \frac{1}{2}\mathbb{E}[f(\mathcal{D}_{i-1}) \mid \boldsymbol{B}_i \neq \boldsymbol{X}_i] - \frac{1}{2} \right| \\
&\quad = \left| \mathbb{E}[f(\mathcal{D}_{i-1}) \oplus \boldsymbol{B}_i \oplus \boldsymbol{X}_i] - \frac{1}{2} \right| \\
&\quad \leqslant \mathbb{E}_{\boldsymbol{B}}\left[\mathbb{E}_{\boldsymbol{X}}\left[\left| f(\mathcal{D}_{i-1}) \oplus \boldsymbol{B}_i \oplus \boldsymbol{X}_i - \frac{1}{2} \right| \right] \right] \leqslant \varepsilon.
\end{aligned}
$$

The last inequality is true because, for any fixing of $\boldsymbol{B}$, if we let

$$
g(\boldsymbol{X}) = f(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{i-1}, \boldsymbol{B}_1, \ldots, \boldsymbol{B}_n) \oplus \boldsymbol{B}_i \oplus \boldsymbol{X}_i,
$$

either g or $\overline{g}$ is checking whether f successfully predicts $\boldsymbol{X}_i$ given $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{i-1}$; so by the ε-next-bit-unpredictability of $\boldsymbol{X}$ for f, we have that $\mathbb{E}_{\boldsymbol{X}}\left[\left|f(\mathcal{D}_{i-1}) \oplus \boldsymbol{B}_i \oplus \boldsymbol{X}_i - \frac{1}{2}\right|\right] \leqslant \varepsilon$.

Therefore

$$
\sum_{i=1}^{n} |\mathbb{E}[f(\mathcal{D}_i) - f(\mathcal{D}_{i-1})]| \leqslant n \cdot \varepsilon,
$$

and so $\boldsymbol{X}$ (εn)-fools f. □

References

[AGHP92] Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures & Algorithms*, 3(3):289–304, 1992. 65

[Ajt83] Miklós Ajtai. 11-formulae on finite structures. *Annals of pure and applied logic*, 24(1):1–48, 1983. 29

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in p. *Annals of mathematics*, pages 781–793, 2004. 9

[Ber41] Andrew C Berry. The accuracy of the gaussian approximation to the sum of independent variates. *Transactions of the american mathematical society*, 49(1):122–136, 1941. 99

[BNS91] László Babai, Noam Nisan, and Márió Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 6–20, 1991. 54

[Bra08] Mark Braverman. Polylogarithmic independence fools ac 0 circuits. *Journal of the ACM (JACM)*, 57(5):1–10, 2008. 78, 81, 87

[BRS91] R. Beigel, N. Reingold, and D. Spielman. The perceptron strikes back. In *[1991] Proceedings of the Sixth Annual Structure in Complexity Theory Conference*, pages 286–291, 1991. 78, 82, 87, 90

[BS90] Ravi B Boppana and Michael Sipser. The complexity of finite functions. In *Algorithms and complexity*, pages 757–804. Elsevier, 1990. 21, 31

[DGJ+10] Ilias Diakonikolas, Parikshit Gopalan, Ragesh Jaiswal, Rocco A Servedio, and Emanuele Viola. Bounded independence fools halfspaces. *SIAM Journal on Computing*, 39(8):3441–3462, 2010. 100, 106

[Dye03] Martin Dyer. Approximate counting by dynamic programming. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 693–699, 2003. 107

[Ess42] Carl-Gustav Esseen. On the liapunoff limit of error in the theory of probability. *Arkiv för matematik, astronomi och fysik*, 28A(9), 1942. 99

[Ess43] Carl-Gustav Esseen. *Determination of the maximum deviation from the Gaussian law*. Almqvist & Wiksell, 1943. 99

[FSS84] Merrick Furst, James B Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical systems theory*, 17(1):13–27, 1984. 29

[Has86] John Hastad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 6–20, 1986. 29, 31, 32

[Has93] Johan Hastad. The shrinkage exponent is 2. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*, pages 114–123. IEEE, 1993. 19

[HH23] Pooya Hatami and William Hoza. Theory of unconditional pseudorandom generators. In *Electron. Colloquium Comput. Complex., TR23-019*, 2023. 4

[IN93] Russell Impagliazzo and Noam Nisan. The effect of random restrictions on formula size. *Random Structures & Algorithms*, 4(2):121–133, 1993. 19

[J+12] Stasys Jukna et al. *Boolean function complexity: advances and frontiers*, volume 5. Springer, 2012. 21, 31

[Jac12] Dunham Jackson. On approximation by trigonometric sums and polynomials. *Transactions of the American Mathematical society*, 13(4):491–515, 1912. 103

[Kan17] Daniel Kane. A structure theorem for poorly anticoncentrated polynomials of gaussians and applications to the study of polynomial threshold functions. *arXiv preprint arXiv:1703.01385*, 2017. 109

[KRW95] Mauricio Karchmer, Ran Raz, and Avi Wigderson. Super-logarithmic depth lower bounds via the direct sum in communication complexity. *Computational Complexity*, 5:191–204, 1995. 21

[LMN93] Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, fourier transform, and learnability. *Journal of the ACM (JACM)*, 40(3):607–620, 1993. 78, 84, 87, 91

[MOO05] Elchanan Mossel, Ryan O'Donnell, and Krzysztof Oleszkiewicz. Noise stability of functions with low influences: invariance and optimality. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 21–30. IEEE, 2005. 110

[MS04] Ben Morris and Alistair Sinclair. Random walks on truncated cubes and sampling 0-1 knapsack solutions. *SIAM journal on computing*, 34(1):195–226, 2004. 107

[MZ10] Raghu Meka and David Zuckerman. Pseudorandom generators for polynomial threshold functions. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, pages 427–436, 2010. 109

[Nec66] Eduard I Neciporuk. A boolean function. In *Soviet Mathematics Doklady*, volume 7, pages 999–1000, 1966. 25

[NN90] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 213–223, 1990. 67

[NW94] Noam Nisan and Avi Wigderson. Hardness vs. randomness. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 2–11, 1994. 110

[O'D14] Ryan O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014. 69

[OW07] Ryan O'Donnell and Karl Wimmer. Approximation by dnf: examples and counterexamples. In *International Colloquium on Automata, Languages, and Programming*, pages 195–206. Springer, 2007. 45

[PZ93] Michael S Paterson and Uri Zwick. Shrinkage of de morgan formulae under restriction. *Random Structures & Algorithms*, 4(2):135–150, 1993. 19

[Ros14] Benjamin Rossman. Formulas vs. circuits for small distance connectivity. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 203–212, 2014. 23

[RTV06] Omer Reingold, Luca Trevisan, and Salil Vadhan. Pseudorandom walks on regular digraphs and the rl vs. l problem. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 457–466, 2006. 9

[Sha49] Claude E Shannon. The synthesis of two-terminal switching circuits. *The Bell System Technical Journal*, 28(1):59–98, 1949. 16

[Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 77–82, 1987. 29, 51

[Spi71] Philip Spira. On time-hardware complexity tradeoffs for boolean functions. In *Proceedings of the 4th Hawaii Symposium on System Sciences, 1971*, pages 525–527, 1971. 23

[Sub61] Bella Abramovna Subbotovskaya. Realization of linear functions by formulas using and, or, and not operations. In *Doklady Akademii Nauk*, volume 136, pages 553–555. Russian Academy of Sciences, 1961. 17, 18

[Tal14] Avishay Tal. Shrinkage of de morgan formulae by spectral techniques. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 551–560. IEEE, 2014. 19

[TS17] Amnon Ta-Shma. Explicit, almost optimal, epsilon-balanced codes. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 238–251, 2017. 67

[Val77] Leslie G Valiant. Graph-theoretic arguments in low-level complexity. In *International Symposium on Mathematical Foundations of Computer Science*, pages 162–176. Springer, 1977. 27

[Vio09] Emanuele Viola. The sum of d small-bias generators fools polynomials of degree d. *Computational Complexity*, 18(2):209–217, 2009. 75, 79

[Yao85] Andrew Chi-Chih Yao. Separating the polynomial-time hierarchy by oracles. In *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, pages 1–10. IEEE, 1985. 29