CS254 Project Report
Fractional Pseudorandom Generators

Winter 2019

1 Introduction

If the study of complexity theory is the study of resource requirements for computation, no survey of complexity theory would be complete without a discussion of randomness as a resource. In modern systems, even when polynomial time deterministic algorithms exist, engineers often deploy randomized algorithms with very small failure probability, simply because the randomized algorithms are so much faster. Randomization is fundamentally required for nearly every cryptographic use, and randomized algorithms can even, in expectation, beat lower bounds for deterministic algorithms in applications like online decision making.

After a random coin has been tossed, a computational device simply has a 1 or a 0 in its memory, just like any other bit on the device. What, then, makes this bit seem more powerful than the deterministic bits that a device can access? Intuitively, some adversary trying to simulate a computation on another machine cannot know the status of this bit. But this is such broad question that to make any progress, we might try to answer a seemingly simpler question: what is the tradeoff, if there is one, between the use of random bits and computational time, particularly in the context of decision problems?

More formally, suppose that L is some language and that D is some deterministic algorithm deciding membership in L. Suppose also that R is some randomized algorithm that, given some x, answers whether or not $x \in L$ correctly at least $2/3$ of the time, and that R is guaranteed to terminate. What is the tradeoff between the running time of A, the running time of any R, and the number of random bits R uses during its execution? In other words, given some R, can R be made into a deterministic algorithm at the cost of some amount of computation time?

As a first attempt, let R be some randomized algorithm that uses k random bits during its execution. A deterministic algorithm could iterate over every possible result of flipping k random coins and feed each result to R in place of random bits, finally taking the majority vote over all the results of R. Because R outputs the correct answer at least $2/3$ of the time, this procedure will output the correct result, but it incurs a multiplicative 2^k increase in computation time. If our deterministic procedure could choose some smaller set of strings, on which A still gives the same results in expectation, then our procedure would run much faster and still give the same result.

More formally, would like to find some set of strings $S \subset \{0,1\}^k$ such that $|E_{y\sim\{0,1\}^k}[A(x,y)] - E_{y\sim S}[A(x,y)]| \leq \varepsilon$ for some small ε. We will say that such an S ε-fools A on x, since, up to ε, A cannot tell whether the y are drawn from

the uniform distribution or from S. Of course, we would like S to work for any input x - for a single x, it is easy to find a very small S. In general, we will talk of fooling classes of algorithms, as fooling a single algorithm can be trivial.

Despite the best attempts of many a complexity theorist, the general question of finding an S that fools all polynomial-time programs remains essentially open. Nevertheless, the difficulty of this problem has not stopped researchers, who instead have chosen to focus on narrower classes of algorithms. For example, Nisan [2] showed how to construct an S of size $n^{O(\log n)}$ for arbitrary space-bounded computation through iterated applications of hash functions. On the other end of the spectrum, when algorithms can only remember one bit, Ta-Shma [4] used random walks on graphs and a variant of Reingold et al's zig-zag graph product [3].

The constructions above and their related literatures have drawn on insights from all across mathematics. Chattopadhyay et al. [1] continue this tradition, cleverly realizing that nearly all of the above results relie on purely discrete mathematics, but that this need not strictly be the case.

A construction of a small set S that fools a class of functions can viewed as a tool for approximating these functions, in that that the probability over $y \in S$ that $A(x, y) = 1$ approximates probability that $A(x, \cdot)$ is 1 over the uniform distribution. But if every function on $\{0,1\}^n$ could be generalized in some coherent way to a function on $[0,1]^n$, then a natural generalization of ε-fooling a function arises; $S \subset [0,1]^n$ fools A on x if $|E_{\{0,1\}^n}[A(x,y)] - E_{y\sim S}[A(x,y)]| \leq \varepsilon$. Relaxing a discrete problem to a continuous analogue has been a successful strategy in many approximation algorithms. For example, a fractional solution to a problem like the Knapsack problem can be much easier to find computationally than a $0-1$ solution. Chattopadhyay et al. show that, similarly here, using this relaxation can make finding good pseudorandom generators much easier, although they stress that they hope that this relaxation tool can be much more useful than just the cases they present.

Note that a derandomization framework could, in theory, look at the details of the function being fooled at the moment and adjust itself on demand. However, an algorithm that unpacks another algorithm and adjusts itself accordingly is very difficult to analyze, and as such most research on derandomization has been focused on finding these sets that ε-fool classes of functions, where the sets can be computed without knowledge of the function to be fooled. As such, any set that ε-fools A on input x for all x will ε-fool all functions $A_x(\cdot) = A(x, \cdot)$. Hence, for the rest of this discussion, we will focus solely on ε-fooling functions on only n bits.

2 Definitions

For simplicity of analysis, we will look at functions on $\{-1, 1\}$, so that the inputs are symmetric about 0, but all of the definitions above can be easily modified in this manner without changing anything substantive.

The first important step towards following the above proof technique is create

a coherent framework for extending functions on $\{-1,1\}^n$ to $[-1,1]^n$. One such extension system that has been well studied is known as the Fourier Transform. For the purposes of this discourse, the important feature of Fourier transform of some function f is that it is a multilinear polynomial that agrees with f on $\{-1,1\}$.

For reference, for any subset $S \subset [n]$, we can define $\hat{f}(S) = 1/2^n \sum_{x \in \{-1,1\}^n} f(x) \Pi_{i \in S} x_i$, and then some analysis shows that if $f'(x) = \sum_{S \subset [n]} \hat{f}(S) \Pi_{i \in S} x_i$, then $f = f'$ on $\{-1,1\}^n$.

Because the Fourier transform is multilinear, we can say that a distribution $S \subset [-1,1]^n$ ε-fools f if $|E_{x \in S}[f(x)] - f(\overline{0})| \leq \varepsilon$.

Trivially, the set $S = \{\overline{0}\}$ satisfies this definition for all functions, but this makes no real progress towards finding a distribution on $\{-1,1\}^n$. In order to be able to round a fractional distribution S to a discrete one, elements of S must at least sometimes differ a little bit from 0.

As such, we will say that S is *p-noticeable* for $0 \leq p \leq 1$ if for all $i \in [n]$, $E_{x \in S}[x_i^2] \geq p$. If an S satisfies this definition, then most of its strings must have most of their indices far from 0. And of course, a 1-noticeable distribution is simply a distribution on $\{-1,1\}^n$.

3 Targeted Classes of functions

As mentioned in the introduction, the space of all Boolean functions is still far too large for researchers to tackle in its entirety. As such, this paper will look at narrower classes of Boolean functions. In particular, the functions analyzed in Chattopadhyay et al [1] and outlined in this paper consist of families of Boolean functions that satisfy the following two properties.

First, a family F of Boolean functions is *closed under restrictions* if, for any $f \in F$, the function f' defined as, for any subset of the inputs, fixing the inputs of f to some constants, is also in F. Many natural classes of functions satisfy this kind of restriction. For example, a Boolean formula where some of the literals are replaced by constants is still a Boolean formula. Or perhaps, if a function f can be computed by a circuit of some fixed depth, then the circuit with certain inputs replaced by constants is a circuit of the same depth that computes a restriction of f.

Second, a family F of Boolean functions satisfies an L_1 *Fourier Tail Bound* with parameters a, b if, for all $f \in F$, $\Sigma_{S \subset [n]:|S|=k} |\hat{f}(S)| \leq a * b^k$. As noted in [5], these bounds are implied by a more well-studied condition, known as the L_2 *Fourier Tail Bound* with parameters a, b, which holds if $\Sigma_{S \subset [n]:|S| \geq k} |\hat{f}(S)| \leq a * 2^{-k/b}$

The first condition is relatively clear in meaning, but the second might appear hard to parse. As noted above, the Fourier expansion of some function f is a multilinear polynomial, where every term has the form $\hat{f}(S) \Pi_{i \in S} x_i$. Note that each term corresponds to some $S \subset [n]$, and thus corresponds roughly to the contribution that a particular interaction of all the variables in S together

make to the output of f, and $\hat{f}(S)$ is the magnitude of this interaction. Informally, then, an L_1 or L_2 Fourier tail bound is a requirement that the value of f is not determined too much by interactions of large numbers of the inputs. For example, the parity function, or XOr, is always dependent on the value of every input, and accordingly its only nonzero Fourier coefficient is $\hat{f}([n]) = 1$. But in a k-junta, a function whose value depends only on k variables, there is a set S of size k such that if some other set T is not a subset of S, then $\hat{f}(T) = 0$. To restate the definition of a k-junta, such a function does not depend at all on interactions between some of the variables in S and some of the variables outside. Informally, if the function has narrow bounds on the Fourier tail, then it cannot look carefully at the entire input, but only at small pieces of it independently, and as a result, one might expect such a function to be easier to fool.[1].

As a final note, if a pseudorandom generator ε-fools every function in some family F of functions closed under restrictions, then clearly the generator fools any subset of the functions F. As such, the first restriction should be thought of as a technical requirement in light of the second condition. If a family of Boolean functions that is not closed under restrictions could be expanded to one that is closed under restrictions that still satsifies the second condition, then the constructions of pseudorandom generators outlined in this paper will still work.

4 Proof Outline

For the full details of the proof, please refer to [1]. What follows is a rough outline of the analysis, with the goal of providing an overview of the main ideas.

4.1 Pseudorandom Generator Construction

Consider the following random walk on $[-1, 1]^n$. Let $X_1, ..., X_t$ be symmetric random variables on $[-1, 1]^n$ drawn from some p-noticeable distribution. For each component j, let $Y_{1,j} = X_{1,j}$ and $Y_{i,j} = Y_{i-1,j} + (1 - Y_{i-1,j})X_{i,j}$ for all $1 < i \leq t$. Then rounding this result into $\{-1, 1\}^n$ will give a pseudorandom generator.

To analyze this, first note that each component of the Y_is is updated independently, and as such we can analyze each component on its own. The more rounding that occurs in the last step, the more error could get introduced in the bias of the output psuedorandom distribution. As such, we might like to show that this walk on one bit polarizes towards -1 or 1. That is to say, we would like to show that for some t (the larger the t, the longer the seed length for the pseudorandom distribution, so we would like t to be small), $Pr(|Y_{i,j}| \geq 1-\delta) \geq 1-\delta$.

In fact, because the $X_{i,j}$s are drawn from a p-noticeable distribution (and thus $E[X_{i,j}^2] \geq p$), the above statement holds for $t = O(\lg(1/\delta)/p)$.

[1] [1] shows exactly this. The smaller the bound on the Fourier tail, the better the parameters of the fooling pseudorandom distribution

Setting $\delta = \epsilon/n$ gives a small enough error for a union bound over all n bits to show that rounding this distribution to $\{-1, 1\}^n$ gives a psuedorandom distribution that ε-fools a family of functions with an L_1 Fourier tail bound (where p depends on the parameters of the tail bound).

A p-stable distribution with good parameters can be directly lifted from constructions of pseudorandom distributions for very narrow classes of functions, scaled from $\{-1, 1\}^n$ to $\{-\beta, \beta\}^n$ for some $0 < \beta < 1$.

As such, what remains to be proved is the fact that this random walk polarizes to $[-1, 1]$.

4.2 Intuition of Construction Analysis

For the detailed proof, see Lemma 3.1 of [1].

To start, note that the construction begins with a p-noticeable fractional psuedorandom generator that ε-fools a family of functions F. We would like the construction to produce a fractional psuedorandom generator with a similar level of error, but that is close to 1-noticeable. If we could show that each step of the walk in the construction did not introduce very much additional error, then we could apply the triangle inequality to show that the construction $O(\varepsilon)$ fools functions in F.

Claim 3.3 (proved below) shows that each step of the walk adds at most ε error, and thus (by Claim 3.4, inducting over steps of the walk), the construction produces a psuedorandom generator with error at most $t\varepsilon$. Given some target error ε_0, this error growth rate is sufficiently slow that we can set ε to be sufficiently small relative to ε_0 and get that $t\varepsilon < \varepsilon_0$.

All that remains to show, then, is that this random walk polarizes towards ± 1. Note that it suffices to show polarization for each bit in the distribution independently.

The full proof (Claim 3.5) is left for the next section, but citechattopadhyay2018psuedorandom gives a good intuition for why polarization occurs. If a random bit is in state Y_i at time i, then its state at time $i + 1$ will differ by at most $|(1 - |Y - i|)X_i|$, which is bounded by $1 - |Y_i|$. As Y_i approaches ± 1, this step bound gets smaller and smaller.

Once a bit gets close to ± 1, it essentially gets trapped and cannot easily move back towards 0. In other words, given the symmetry assumption on the X_is, from any fixed $p \neq 0$, it takes more steps in expectation to walk from p to 0 than it takes to walk from 0 to p.

Putting these all together shows that the random walk produces a psuedorandom generator that $O(\varepsilon)$ fools a family of functions and, for each element Y produced by the generator, every index Y_i is extremely close to ± 1.

5 Conclusion

This paper gives a construction for psuedorandom generators for Boolean functions with bounded L_1 Fourier tails. Many classes of functions satisfy this

requirement, although there are many open questions regarding proving better and better bounds on these Fourier tails. Chattopadhyay et al. compare their generators to existing generators and point out that for some classes of functions, their generator beats the state of the art, but sometimes it is far behind. However, the generator outlined in this paper is the same (up to different choices of parameters) no matter to what class of functions it is applied, and as such does not particularly leverage any distinct properties of the function class. For comparison, many state of the art generator constructions depend heavily on the exact structure of the functions being analyzed.

Moreover, if the Fourier tail bounds of any class of functions are ever improved, this construction automatically will give better psuedorandom generators. And beyond this, the relatively general fractional generator relaxation that the authors give in this work has the potential, the authors hope, to be used as a framework for constructions of psuedorandom generators in many other situations.

6 Appendix: Interesting Omitted Proofs

6.1 Proof of Error Bound (Claim 3.3)

We would like to prove that taking one step in our random walk introduces at most ε error to the psuedorandom distribution. Most of this section is drawn heavily from [1], but we include it here to give an example of why the technical requirement that a family of functions $\mathcal{F}$ be closed under restrictions is critical for the analysis.

Notationally, set δ_y to be $(\delta_y)_i = 1 - |y_i|$, and let $x \circ y$ for two vectors x and y be defined as their coordinatewise product. Let X be some fractional psuedorandom generator for $\mathcal{F}$ with bias ε.

Note that $y + \delta_y \circ X$ is the expression that corresponds to taking one step in the random walk of the psuedorandom generator construction. As such, we would like to show that for any fixed $y \in [-1, 1]^n$ and any $f \in \mathcal{F}$, $|f(y) - E_X[f(y + \delta_y \circ X)]| \leq \varepsilon$.

For any function $f \in \mathcal{F}$, consider the following random restriction of f. Define $F(x) = f(R(x))$ where $R(x)_i = sign(y_i)$ with probability $|y_i|$ and x_i otherwise. Because the Fourier extension of f is a multilinear function and because R acts independently on each coordinate, $E_F(F(x)) = E_R(f(R(x)) = f(E_R(R(x))) = f(y + \delta_y \circ x)$. By similar logic, note that $f(y) = E_F(F(\overline{0})$.

Taking the expectation over $x \in X$ gives the expression that we with to bound, namely, $|f(y) - E_X(f(y + \delta_y \circ x))| = |E_F(F(\overline{0}) - E_{F,X}(F(X))| \leq E_F|F(\overline{0}) - E_X(F(X))|$.

However, X ε-fools all $f \in \mathcal{F}$. This is where the fact that $\mathcal{F}$ is closed under restrictions comes into play. Because every restriction $F \in \mathcal{F}$, then X ε-fools a randomly chosen restriction. Hence $|F(\overline{0}) - E_X(F(X))| \leq \varepsilon$, which implies $E_F|F(\overline{0}) - E_X(F(X))| \leq \varepsilon$.

6.2 Proof of Polarization (Claim 3.5)

We would like to prove that the random walk of the construction on one bit polarizes to ± 1 quickly. Again, most of this section is drawn heavily from [1], but we include it here for completeness - this proof is key to the entire construction - and because random walks that polarize to some set of values, instead of converging to one value, are somewhat rare and interesting in their own right.

For simplicity of notation, let $X_1, ..., X_t$ be the strings drawn from a p-noticeable distribution, and let $Y_1 = X_1$ and $Y_i = Y_{i-1} + (1 - |Y_{i-1}|)X_i$. Let $Z_i = 1 - |Y_i|$, or in other words, the distance remaining between Y_i and ± 1. We would like to prove, then, that $Pr[Z_t \geq \delta]$ is very small. Note that it is equivalent to show that $E[1 - Y_t^2]$ is very small (equivalently, Y_t is q-noticeable for q very close to 1).

Some simple computation shows that $Z_i \leq Z_{i-1}(1 - X_i)$. Z_i and X_i are independent variables, so $E[\sqrt{Z_i}] \leq E[\sqrt{Z_{i-1}}]E[\sqrt{1 - X_i}]$.

The Taylor expansion of $\sqrt{1 - x}$ on $[-1, 1]$ has coefficients that are all negative. Moreover, because X_i is a symmetric random variable, all of the odd degree terms cancel to 0 in expectation. Then $E[\sqrt{1 - X_i}] \leq 1 - E[X_i^2]/8 \leq 1 - p/8 \leq exp(-p/8)$.

This bound, combined with the above recurrence, gives that $E[\sqrt{Z_t}] \leq exp(-tp/8)$. An application of Markov's inequality and some rearranging gives the final desired result.

References

[1] Eshan Chattopadhyay, Pooya Hatami, Kaave Hosseini, and Shachar Lovett. Pseudorandom generators from polarizing random walks. In *33rd Computational Complexity Conference (CCC 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[2] Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

[3] Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders. *Annals of mathematics*, pages 157–187, 2002.

[4] 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. ACM, 2017.

[5] Avishay Tal. Tight bounds on the fourier spectrum of ac0. In *32nd Computational Complexity Conference (CCC 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.