COMS 6998: Unconditional Lower Bounds and Derandomization **Spring 2024**

Lecture 12: April 9, 2024

Lecturer: Rocco Servedio *Scribes: Jiaze Xu*

1 Deterministic Approximate Counting for LTFs

Definition 1 (Relative Error Deterministic Approximate Counting for LTFs)**.** *Let $f(x) = \text{sign}(w \cdot x - \theta)$ be an n-variable LTF. Let $\epsilon > 0$, and let N be the number of satisfying assignments of f. On input w, θ, ϵ, deterministic approximate counting outputs an integer $\hat{N}$ such that $N \leqslant \hat{N} \leqslant (1+\epsilon)N$.*

The first **randomized** algorithm to solve this problem dates back 1999. We now propose a **deterministic** version as follows:

Theorem 2. *Following definition 1, suppose $\forall i \in [n]$, $w_i \in \mathbb{Z}$ and $\theta \in \mathbb{Z}$. Let $W = \max(\{|w_i| : i \in [n]\} \cup \{|\theta|\})$. Then there exists a $\text{poly}(n, \log(W), \frac{1}{\epsilon})$-time deterministic algorithm solving the determinisitic approximate counting problem. [GKM10]*

The high-level idea of the proof is to:

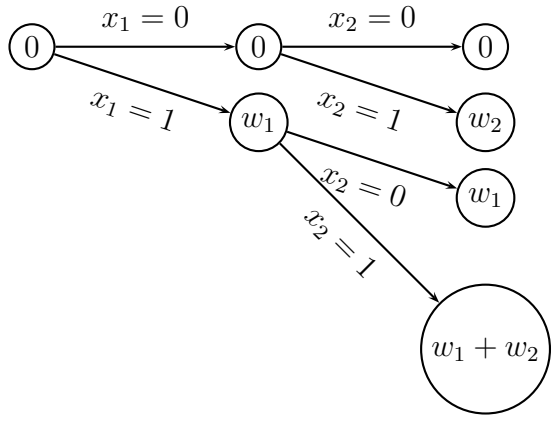
- approximate the LTF using a branching program (to be defined below)
- use standard dynamic programming to count the exact number of satisfying assignments to the branching program

Definition 3 (branching program)**.** *An (S, T)-branching program ((S, T)-BP) is a layered directed acyclic graph corresponding to a function $f : \{0,1\}^T \to \{0,1\}$, where:*

- *there are $T+1$ layers, labeled $0, 1, ..., T$*
- *on each layer, there are at most S vertices ("states")*
- *layer 0 contains a single source vertex s*
- *for each layer i where $i \in \{0, ..., T-1\}$, for each vertex v_1 in layer i, there's an edge from v_1 to some vertex v_2 in layer $i+1$, and another edge from v_1 to some vertex v_3 in layer $i+1$, corresponding to the cases where variable x_{i+1} is assigned 0 or 1.*

- *each vertex in layer T is labeled either 0 (reject) or 1 (accept)*

The branching program intuitively calculates a boolean function. In particular, an LTF $f(x) = \text{sign}(w \cdot x - \theta)$, where $\forall i \in [n], w_i \in \mathbb{Z}_{\geqslant 0}$ and $\theta \in \mathbb{Z}_{\geqslant 0}$, can be computed as follows: Let $W = \sum_{i=1}^{n} w_i$. We define a $(W+1, n)$-BP, where each state in layer j corresponds to the possible prefix sums up to index j: $\sum_{i=1}^{j} w_i x_i$. We accept or reject based on whether the corresponding node in the final layer is reachable. Below is an illustration of the first 3 layers:



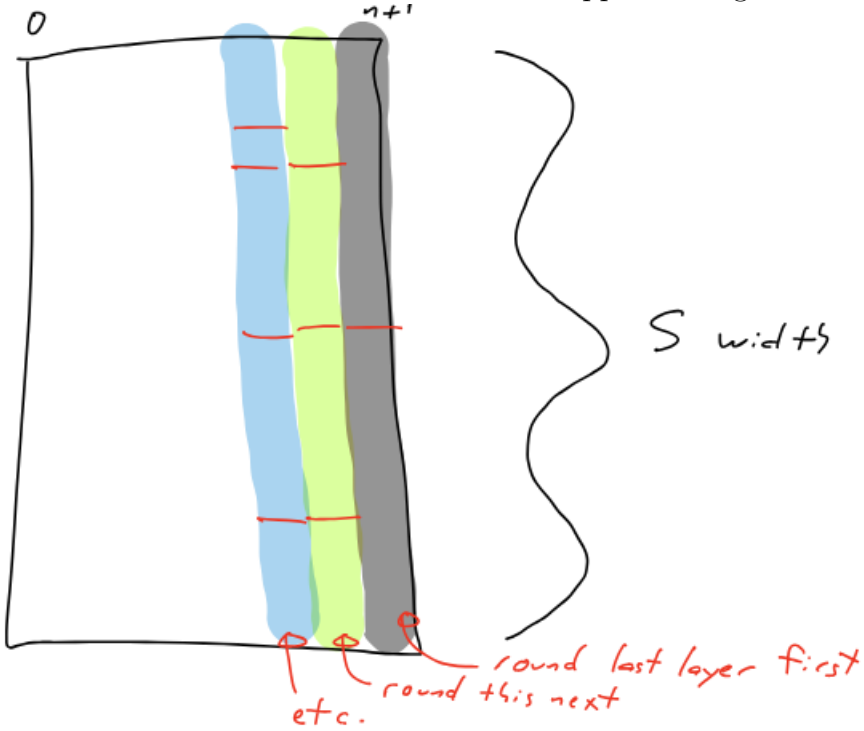
Fact 4. *We can compute the number of satisfying assignments of an (S, T)-BP in $\text{poly}(S, T)$-time, using dynamic programming.*

Now, we approximate the $(W+1, n)$-BP for our LTF with a $(\text{poly}\log(W), n+1)$-BP. The states of our new BP corresponds to subsets of the states of the original BP, gained via a process called **rounding**:

We partition the states in layer i of the original BP into intervals $I_1 = [0, v_1 - 1], I_2 = [v_1, v_2 - 1], ..., I_t = [v_{t-1}, W]$, such that the number of accepting suffixes for all the partial sums in an interval is roughly the same (in other words, the probability of elements in each interval reaching the accept state is roughly the same). In particular, when the acceptance probability drops by a factor of $1 + \epsilon$, we start a new interval. $I_1, ..., I_t$ constitutes the new states in layer $i + 1$ of our new BP.

The challenge is to calculate the probabilities of reaching acceptance state, which is itself an instance of approximate counting of number of satisfying assignments of an LTF. To solve this, we again use dynamic programming, but this time from back to front (i.e. we round layer $n + 1$ first, and then layer n, util reaching layer 0). This is because the probability of a prefix sum in layer i reaching acceptance state can be calculated by combining the probability of a prefix sum in layer $i + 1$ reaching acceptance state and by the value of x_i, which naturally defines a recurring problem.

An illustration of the "back to front" approach is given below:



Note that solving the non-negative integer version of the problem automatically solves the integer version of the problem: the possible sums range from $-\sum_{i=1}^{n}|w_i| - |\theta|$ to $\sum_{i=1}^{n}|w_i| + |\theta|$, which can be expressed with $\max(\{|w_i| : i \in [n]\} \cup \{|\theta|\})$ bits.

2 Deterministic Approximate Counting for PTFs

Definition 5 (PTF). *A degree-d PTF is a boolean function $f : \{-1,1\}^n \to \{-1,1\}$ such that $f(x) = \text{sign}(p(x))$, where p is a degree-d real polynomial.*

Meka and Zuckerman proposed a PRG for degree-d PTFs with seed length $(d/\epsilon)^{O(d)} \log(n)$ [MZ13]. By enumerating over all seeds, this gives an approximate counting algorithm running in $n^{(d/\epsilon)^{O(d)}}$ time. Another PRG is proposed with seed length $O_d(1)\,\text{poly}(1/\epsilon)\log(n)$ seed length, and this yields an approximate counting algorithm running in $O_d(1)\,\text{poly}(1/\epsilon)$ time [Kan17].

It turns out that this is a problem for which we can do faster deterministic approximate counting than just enumerating over all seeds of a PRG: it is known that we can do $O(d,\epsilon)(1)$-time deterministic approximate counting for degree-d PTFs. Note that

the input size of degree-d PTF is roughly n^d, so for constant d this time complexity is polynomial in input size.

2.1 $d = 2$

We draw special attention to the $d = 2$ case and provide an overall sketch. The high level idea is to extend Berry-Esseen theorem from the previous lecture. The insight of Berry-Esseen is that regular linear formulas on independent random variables behave like a Gaussian. Using the "invariance principle" proposed by Mossel et al. [MOO05], we can extend Berry-Esseen to low-degree polynomials.

Let $p(x)$ be a "regular" polynomial (similar to the "regular" LTFs defined in the previous lecture). The goal is then to transfer the distribution of $p(\mathbf{x}_1, ..., \mathbf{x}_n)$, where $\mathbf{x} \sim \{\pm 1\}^n$, to the distribution of $p(\mathbf{g}_1, ..., \mathbf{g}_n)$, where $g \sim \mathcal{N}(0, 1)^n$, and analyze the polynomials with Gaussian inputs.

3 Nisan-Widgerson PRG

3.1 High level overview

The Nisan-Widgerson (NW) PRG is a generic way of constructing a PRG. The high level idea is that given an average-case lower bound against a "richer" class $\mathcal{C}'$ of r-variable boolean functions, we are able to construct a PRG for a "simplier" class $\mathcal{C}$ of n-variable boolean functions.

On a high level, the NW generator G takes s truly-random bits $\mathbf{U} = \mathbf{U}_1, ..., \mathbf{U}_s$ as input, and outputs a n-bit pseudorandom string. It works as follows:

Let $\mathbb{S} = (S_1, ..., S_n)$, where $\forall i \in [n], S_i \subseteq [s]$ and $|S_1| = ... = |S_n| = r$. We make it such that for each $i, j \in [n]$ where $i \neq j$, $|S_i \cap S_j|$ is very small. Let $h : \{0,1\}^r \to \{0,1\}$ be average-case hard for $\mathcal{C}'$. The NW generator is:

$$G(\mathbf{U}) = (h(\mathbf{U}|_{S_1}), ..., h(\mathbf{U}|_{S_n})). \tag{*}$$

The intuition is that the hardness of h makes each bit of the output unpredictable, which shows a "hardness vs randomness" intuition.

Now we define the class $\mathcal{C}'$ as a composition of $\mathcal{C}$ and Juntas. Let k be the supremum of $|S_i \cap S_j|$ where $i \neq j$. Let $\text{Junta}_{r,k}$ be the set of functions $f : \{0,1\}^r \to \{0,1\}$ such that f is a k-junta.

$$\mathcal{C}' = \mathcal{C} \circ \text{Junta}_{r,k} = \{f(g_1(x), ..., g_n(x)) : f \in \mathcal{C}, \forall i \in [n], g_i \in \text{Junta}_{r,k}\}.$$

3.2 Preliminary theorem

Before introducing the NW theorem, consider the following preliminary version:

Theorem 6 (Preliminary NW theorem)**.** *Let $\mathcal{C}$ be a class of functions $\{0,1\}^n \to \{0,1\}$. Let h be ϵ-hard for $\mathcal{C}' = \mathcal{C} \circ \text{Junta}_{r,k}$. Let $\mathbb{S} = (S_1, ..., S_n)$ be an (s,k,r)-design (to be defined in definition 7). Then the generator (*) (ϵn)-fools $\mathcal{C}$ and has seed length s.*

We define and prove the existence of (s,k,r)-design below:

Definition 7 ((s,k,r)-design)**.** *A list $\mathbb{S} = (S_1, ..., S_n)$ of n subsets $S_i \subset [s]$ is an (s,k,r)-design if*

1. $\forall i \in [n], |S_i| = r$
2. $\forall i \neq j, |S_i \cap S_j| \leqslant k$

We would want the following properties:

- $s << n$, since the seed length should be small
- k is very small, since large k implies more powerful $\text{Junta}_{r,k}$ and therefore more powerful $\mathcal{C}'$, making it hard to find a good h

However, we notice the following tensions among the parameters:

- small k leads to large s
- $s \geqslant \log(n)$ by definition of seed

Fortunately, we're able to construct a design following a greedy algorithm:

Lemma 8. *Let $c \geqslant 1, s = 100c^2 \log(n), r = c\log(n), k = \log(n)$. There exists a greedy algorithm that constructs $\mathbb{S}$ satisfying definition 7 that runs in $\leqslant \text{poly}(n) \cdot 2^s$ time.*

Proof. The greedy algorithm runs in n stages: it tries all $|S_i| = r$ at stage i, looking for one such that $\forall j < i$, $|S_i \cap S_j| \leqslant k$. It FAILs if no such sets exist.

The proof for time complexity is trivial. We'll show this algorithm never fails:

After picking sets $S_1, ..., S_{i-1}$, a random $\mathbf{S} \subset [s]$ where $|\mathbf{S}| = r$ has the following property:

$$\Pr_{\mathbf{S}}[\exists j \in [i-1], |\mathbf{S} \cap S_j| > k] < 1.$$

which follows from showing:

$$\Pr_{\mathbf{S}}[|\mathbf{S} \cap \{1, ..., r\}| > k] < \frac{1}{n}.$$

But

$$\begin{aligned}
&\mathbb{E}_{\mathbf{S}}[|\mathbf{S} \cap \{1, ..., r\}|] \\
=&\mathbb{E}_{\mathbf{S}}[|\mathbf{S} \cap \{1, ..., c\log(n)\}|] \\
=&c\log(n)\frac{c\log(n)}{100c^2\log(n)} \\
=&\frac{\log(n)}{100} \\
=&\frac{k}{100}.
\end{aligned}$$

This suggests that it is very unlikely that $|\mathbf{S} \cap \{1, ..., r\}| > k$. Using the multiplicative version of Chernoff bound for negatively correlated random variables, or equivalently, calculating directly with binomial coefficients, we can prove our result. ■

3.3 Actual NW Theorem

Now it's time to introduce the actual NW theorem:

Theorem 9 (NW theorem)**.** *Let $\mathcal{C}$ be a class of functions $\{0,1\}^n \to \{0,1\}$. Let h be ϵ-hard for $\mathcal{C}' = \mathcal{C} \circ \text{Junta}_{r,k}$. Let $\mathbb{S} = (S_1, ..., S_n)$ be an (s, k, r)-design where $c \geqslant 1, s = 100c^2\log(n), r = c\log(n), k = \log(n)$. Then the generator (*) (ϵn)-fools $\mathcal{C}$ and has seed length s [NW88].*

Before proving the theorem, we first look at an application: fooling $\mathcal{AC}^0$ circuits.

Corollary 10. *For $M \geqslant n$, the NW generator gives a δ-PRG for $\mathcal{AC}^0_{M,d}$ with seed length $s = (\log(M/\delta))^{2d+O(1)}$, computable in $\text{poly}(n) \cdot 2^s$ time.*

Proof. With proper parameter setting and our earlier average case lower bound against $\mathcal{AC}^0$, using $\mathcal{AC}^0_{M,d} \circ \text{Junta}_{r,k}$ is contained in $\mathcal{AC}^0_{M+n\cdot 2^k, d+2}$. ■

Now we prove theorem 9. The key notion is "next-bit unpredictability":

Definition 11. *Let* $\mathbf{X}$ *be a random variable over* $\{0,1\}^n$*. Let* $f : \{0,1\}^n \to \{0,1\}$*. Let* $\epsilon > 0$*. We say* $\mathbf{X}$ *is* ϵ*-next-bit-unpredictable for* f *(*ϵ*-nbu) if for each* $i \in [n]$*, for each* $a \in \{0,1\}^{n-i+1}$*, we have*

$$|\Pr_{\mathbf{X}}[f(\mathbf{X}_1, ..., \mathbf{X}_{i-1}, a) = \mathbf{X}_i] - \frac{1}{2}| \leqslant \epsilon.$$

equivalently, $\mathbf{X}$ ϵ*-fools* $x \mapsto f(x_1, ..., x_{i-1}, a) \oplus x_i$*.*

We can therefore decompose theorem 9 into the following lemmas:

Lemma 12. *NW generator, under setup of theorem 9, is* ϵ*-nbu for all* $f \in \mathcal{C}$*.*

Lemma 13. *Let* $\mathbf{X}$ *be a random variable over* $\{0,1\}^n$*. Let* $f : \{0,1\}^n \to \{0,1\}$*. If* $\mathbf{X}$ *is* ϵ*-nbu for all* $f \in \mathcal{C}$*, then* $\mathbf{X}$ ϵn*-fools every* $f \in \mathcal{C}$*.*

To prove the first lemma:

Proof. Fix any $f : \{0,1\}^n \to \{0,1\}$ in $\mathcal{C}$. Fix any $i \in [n]$, $a \in \{0,1\}^{n-i+1}$. Let $\mathbf{U} \sim \mathcal{U}_s$, $\mathbf{X} = G(\mathbf{U})$. We have:

$$\begin{aligned}
&|\Pr_{\mathbf{X}}[f(\mathbf{X}_1, ..., \mathbf{X}_{i-1}, a) = \mathbf{X}_i] - \frac{1}{2}| \\
=&|\mathbb{E}_{\mathbf{U}_{[s]\setminus S_i}}[\Pr_{\mathbf{U}_{S_i}}[f(h(\mathbf{U}|_{S_1}), ..., h(\mathbf{U}|_{S_{i-1}}), a)]] - \frac{1}{2}| \\
\leqslant&\mathbb{E}_{\mathbf{U}_{[s]\setminus S_i}}|[\Pr_{\mathbf{U}_{S_i}}[f(h(\mathbf{U}|_{S_1}), ..., h(\mathbf{U}|_{S_{i-1}}), a)]] - \frac{1}{2}|.
\end{aligned}$$

For each fixing of $\mathbf{U}_{[s]\setminus S_i}$, write $\mathbf{Z} = \mathbf{U}_{S_i}$. For each $j < i$, since we fixed $\mathbf{U}_{[s]\setminus S_i}$ and $|S_i \cap S_j| \leqslant k$, there's a k-junta g_j such that $h(\mathbf{U}|_{S_i}) = g_j(\mathbf{Z})$. So

$$\begin{aligned}
&|[\Pr_{\mathbf{U}_{S_i}}[f(h(\mathbf{U}|_{S_1}), ..., h(\mathbf{U}|_{S_{i-1}}), a)]] - \frac{1}{2}| \\
=&|[\Pr_{\text{Unif } \mathbf{Z}}[f(h(g_1(\mathbf{Z})), ..., h(g_{i-1}(\mathbf{Z})), a)]] - \frac{1}{2}| \\
\leqslant&\epsilon.
\end{aligned}$$

The last inequality is because h is ϵ-hard for $\mathcal{C} \circ \text{Junta}_{r,k}$. ■

To prove the second lemma, we use the hybrid argument:

Proof. Let $\mathbf{B} = (\mathbf{B}_1, ..., \mathbf{B}_n) \sim \mathcal{U}_n$. Consider the following hybrid distributions over $\{0,1\}^n$:

$$\begin{aligned}
&\mathbf{D}_0 = (\mathbf{B}_1, ..., \mathbf{B}_n) = \mathbf{B}\\
&\mathbf{D}_1 = (\mathbf{X}_1, \mathbf{B}_2, ..., \mathbf{B}_n)\\
&\mathbf{D}_2 = (\mathbf{X}_1, \mathbf{X}_2, \mathbf{B}_3, ..., \mathbf{B}_n)\\
&...\\
&\mathbf{D}_i = (\mathbf{X}_1, ..., \mathbf{X}_i, \mathbf{B}_{i+1}, ..., \mathbf{B}_n)\\
&...\\
&\mathbf{D}_n = (\mathbf{X}_1, ..., \mathbf{X}_n) = \mathbf{X}
\end{aligned}$$

By triangular inequality:

$$\begin{aligned}
&|\mathbb{E}[f(\mathbf{X})] - \mathbb{E}[f(\mathbf{D})]|\\
=&|\mathbb{E}[f(\mathbf{D}_n)] - \mathbb{E}[f(\mathbf{D}_0)]|\\
\leqslant&\sum_{i=1}^{n} |\mathbb{E}[f(\mathbf{D}_i)] - \mathbb{E}[f(\mathbf{D}_{i-1})]|. \qquad (**)
\end{aligned}$$

Fix $i \in [n]$. We have:

$$\begin{aligned}
&|\mathbb{E}[f(\mathbf{D}_i)] - \mathbb{E}[f(\mathbf{D}_{i-1})]|\\
=&|\mathbb{E}[f(\mathbf{D}_i)|\mathbf{B}_i = \mathbf{X}_i] - (\frac{1}{2}\mathbb{E}[f(\mathbf{D}_{i-1})|\mathbf{B}_i = \mathbf{X}_i] + \frac{1}{2}\mathbb{E}[f(\mathbf{D}_{i-1})|\mathbf{B}_i \neq \mathbf{X}_i])|\\
=&|\mathbb{E}[f(\mathbf{D}_i)|\mathbf{B}_i = \mathbf{X}_i] - \frac{1}{2}\mathbb{E}[f(\mathbf{D}_i)|\mathbf{B}_i = \mathbf{X}_i] - \frac{1}{2}\mathbb{E}[f(\mathbf{D}_{i-1})|\mathbf{B}_i \neq \mathbf{X}_i]|\\
=&|\frac{1}{2}\mathbb{E}[f(\mathbf{D}_i)|\mathbf{B}_i = \mathbf{X}_i] + \frac{1}{2}\mathbb{E}[\overline{f(\mathbf{D}_i)}|\mathbf{B}_i \neq \mathbf{X}_i] - \frac{1}{2}|\\
=&|\mathbb{E}[f(\mathbf{D}_i) \oplus \mathbf{B}_i \oplus \mathbf{X}_i] - \frac{1}{2}|\\
=&\mathbb{E}_{\mathbf{B}}[|\mathbb{E}_{\mathbf{X}}[f(\mathbf{D}_i) \oplus \mathbf{B}_i \oplus \mathbf{X}_i] - \frac{1}{2}|].
\end{aligned}$$

Note that from the third to the fourth equality, we utilize the fact that for any p, $-p = (1-p) - 1$.

For any fixing of $\mathbf{B}$, if we let $g(x) = f(x_1, ..., x_{i-1}, \mathbf{B}_i, ..., \mathbf{B}_n) \oplus \mathbf{B}_i \oplus x_i$, either g or $\overline{g}$ is checking whether f successfully predicts x_i given $x_1, ..., x_{i-1}$. So by ϵ-nbu of f for $\mathbf{X}$, the above is $\leqslant \epsilon$. Hence $(**) \leqslant \epsilon n$, i.e. $\mathbf{X}$ ϵ-fools f. ■

References

[GKM10] Parikshit Gopalan, Adam Klivans, and Raghu Meka. Polynomial-time approximation schemes for knapsack and related counting problems using branching programs. In *Data Structures and Algorithms*, 2010. 2

[Kan17] Daniel Kane. A structure theorem for poorly anticoncentrated polynomials of Gaussians and applications to the study of polynomial threshold functions. *The Annals of Probability*, 45(3):1612–1679, 2017. 2

[MOO05] Elchanan Mossel, Ryan O'Donnell, and Krzysztof Oleszkiewicz. Noise stability of functions with low influences: invariance and optimality. In *Proc. 46th IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society Press, 2005. 2.1

[MZ13] Raghu Meka and David Zuckerman. Pseudorandom generators for polynomial threshold functions. *SIAM Journal on Computing*, 42(3), 2013. 2

[NW88] Noam Nisan and Avi Wigderson. Hardness vs. randomness. In *Proc. 29th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 2–11. IEEE Computer Society Press, 1988. 9