6.842 Randomness and Computation February 28, 2020

Lecture 7

Lecturer: Ronitt Rubinfeld *Scribe: Steven Qu*

Overview

More applications of pairwise independence:

1. Reducing algorithmic error probability
2. Interactive proof for finding lower bound of set size

Recall that last time we covered how to use pairwise independence to "shrink" the size of functions.

1 Reducing error

Given an algorithm $\mathcal{A}$ for some language $L \in \mathsf{RP}$ which uses m random bits R, we know that by definition:

$$\begin{cases} \text{if } x \in L, & Pr[\mathcal{A}(x,R) = \text{“accept”}] > 1/2 \\ \text{if } x \notin L, & Pr[\mathcal{A}(x,R) = \text{“accept”}] = 0. \end{cases}$$

Note: The setup here assumes one-sided error, but the same methods will also work for two-sided error.

1.1 Repeating $\mathcal{A}$

Our first method of reducing error is by repeating $\mathcal{A}$, k times, with new random bits each time. Due to the one-sided nature of $\mathcal{A}$, we can safely output accept if any of these runs accepts, and otherwise reject. This method has the following behavior:

$$\begin{cases} \text{if } x \in L, & Pr[\text{“accept”}] \geq 1 - 1/2^k \\ \text{if } x \notin L, & Pr[\text{“accept”}] = 0. \end{cases}$$

The new error probability is exponentially less in k, and the total number of random bits we use is km. What if we wanted to use fewer random bits?

1.2 Using Pairwise Independence

As we saw last class, we can use pairwise independence to reduce the number of random bits we need. Using something analogous to the "two-point sampling" method, we can generate $\mathcal{H}$, a family of pairwise independent functions (or equivalently, a 2-universal hash family), each one mapping $[2^{k+2}] \to \{0,1\}^m$ such that only $O(k+m)$ random bits are required to pick a specific $h \in \mathcal{H}$ in $\mathsf{poly}(k,m)$ time.

Now, consider the following new-and-improved sampling algorithm $\mathcal{B}$:

1: **procedure** $\mathcal{B}(x, \mathcal{H})$
2: pick some function h randomly from $\mathcal{H}$
3: **for** $i \in [2^{k+2}]$ **do**
4: Check $\mathcal{A}(x, h(i))$
5: if ever accept, output "accept"; otherwise reject.

First, note that line 2 is the only place in $\mathcal{B}$ that uses randomness. Also notice that the runtime, due to the for loop, is $O(2^{k+2} \cdot \text{runtime of } \mathcal{A})$.

Now, if $x \notin L, Pr[\text{"accept"}] = 0$ still. Now, if $x \in L$, then we misclassify if and only if we never see accept for all $i \in [2^{k+2}]$. Let $r_i = h(i)$ and $\sigma_x(r_i) = \mathbb{1}[\mathcal{A}(x, r_i) \text{ accepts}]$. Then, $\mathcal{B}$ accepts if and only if $Y = \sum_i \sigma_x(r_i) > 0$. We note that $\mathbb{E}[Y/2^{k+2}] = \mathbb{E}[\sigma_x(r_i)] > 1/2$.

Now, recall two useful lemmas:

Lemma 1. *(Chebyshev's inequality.) Given random variable X with finite expected value, we have that*

$$Pr[|X - \mathbb{E}X| \geq \epsilon] \leq \text{Var}(X)/\epsilon^2.$$

When we assume X is a sum of pairwise independent variables $X_i \in [0, 1]$, a stronger statement can be made:

Lemma 2. *(Pairwise independent tail inequality.) Given $X_1, X_2, ..., X_t \in [0, 1]$ pairwise independent with $X = \frac{1}{t}\sum_i X_i$, then we have that*

$$Pr[|X - \mathbb{E}X| \geq \epsilon] \leq \frac{1}{t\epsilon^2}.$$

Note that this lemma assumes a weaker condition than the Chernoff bounds, which require mutual (complete) independence.

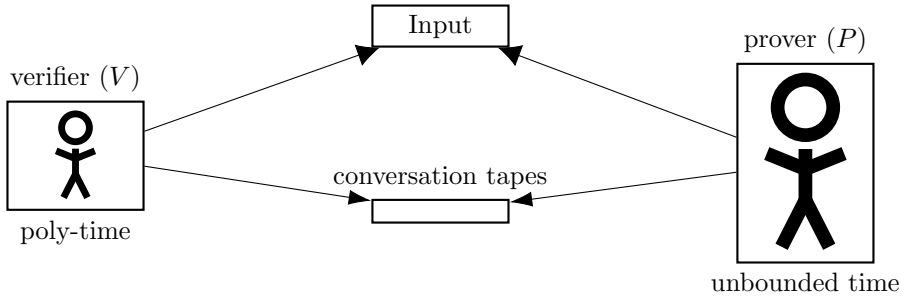
Applying Lemma 2 on Y, we have that if $x \in L$, then

$$Pr[\mathcal{B} \text{ rejects}] = Pr[Y/2^{k+2} = 0] \leq \frac{1}{2^{k+2}(\frac{1}{2})^2} = \frac{1}{2^k}.$$

2 Lower Bounding Set Sizes

2.1 Interactive Proofs

The basic idea of an interactive proof is that a powerful solver interacts with you to convince you that it "knows" a proof of some statement (w.h.p.), thus implying that the proof exists and that the statement must be true. (This idea, unsurprisingly, originates from cryptography.) An example is the Coke-Pepsi challenge: to convince you that I know the difference between the two drinks (or to convince you that there is a difference at all), it is not necessary for me to reveal any hidden information; rather, you can repeatedly challenge me to identify a randomly chosen drink, and if I can correctly do this at a much better rate than random guessing, then it is likely I can tell the difference.



Noting the setup of an interactive proof above, we have the following definition:

Definition 3. *Interactive Proof System for language L consists of a verifier and a prover, with access to the same input and can communicate to each other. Given x, there exists a protocol such that:*

- *if both V and P follow the protocol (V has randomness; P doesn't need it) and $x \in L$, then $Pr_V[V \text{ accepts } x] \geq 2/3$;*
- *if V follows the protocol and $x \notin L$, then $Pr_V[V \text{ accepts } x] \leq 1/3$.*

P wants to convince V to accept x, regardless of whether $x \in L$. Per the definition, when $x \in L$, it is better for P to follow the protocol. However, when $x \notin L$, P may play adversarially and not follow the protocol. In this case, we want V to be able to reject x with a good probability regardless of what P does.

Definition 4. IP *is the class of languages L with an interactive proof system where V runs in polynomial time.*

Note: Based on the Coke-Pepsi example, it seems like it would be useful for V's random bits to be private from P. It turns out that forcing V to use only public bits (as in, it shares all of its random bits with P) provides the same amount of power.

2.2 Satisfiability Set

Problem: Suppose we wanted to determine the number of satisfying assignments for a given boolean formula. More precisely, given a boolean formula φ, let S_φ be the set of assignments x that satisfy φ. Suppose that for a given x, V can check if $x \in S_\varphi$. Then, we have the following claim about $|S_\varphi|$:

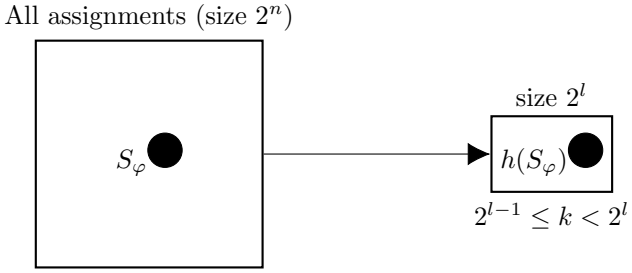
Claim 5. *There exists a protocol such that, on input φ:*

- *if $|S_\varphi| > k$ and both V and P follow the protocol, then $Pr_V[V \text{ accepts } \varphi] \geq 2/3$;*
- *if $|S_\varphi| < k/\Delta$ (for some fixed Δ) and V follows the protocol, then $Pr_V[V \text{ accepts } \varphi] \leq 1/3$.*

In other words, the problem of deciding whether the size of some boolean formula's satisfying set is lower bounded by some value k (or upper bounded by some fixed constant factor of k) is in IP.

Suppose there are n variables in the boolean formula. Without P entirely, we could just use V to randomly sample from all assignments, and check if they satisfy the formula. However, this is bad if k is small relative to 2^n, in which case it would take exponential time to determine the lower bound with a good probability.

The idea is then to map the set of all assignments onto a smaller set, with size on the order of k. V can't exactly sample from this set, because it might not know how to invert the mapping, but it can tell P to do this. In fact, because this smaller set has size on the order of k, it is enough to see if any one element, say the string of all 0s, has a preimage. P will invert this sample to a satisfying assignment if it exists, and all V needs to do is to verify this.



We need that $|h(S_\varphi)| \approx |S_\varphi|$. This we will prove later. For now, consider the following protocol. Find l such that $2^{l-1} \leq k < 2^l$. Then, given $\mathcal{H}$, a family of pairwise independent functions (generated as before) mapping $\{0,1\}^n \to \{0,1\}^l$:

1. V picks a function h at random from $\mathcal{H}$.
2. V tells P this function h.
3. P tells V $x \in S_\varphi$ such that $h(x) = 0^l$.
4. V accepts if and only if $x \in S_\varphi$ and $h(x) = 0^l$.

Some small notes: Similar to before, step 1 is the only randomness used in the protocol. In step 2, if $\mathcal{H}$ had been the set of all mutually independent functions, then h would have been too large to be describable in polynomial time. This is why pairwise independence is important. In step 3, we can use any string in $\{0,1\}^l$, not just 0^l.

The basic idea behind the protocol is that, if $|S_\varphi| > k$, then something in the set will probably map to 0^l; on the other hand, if $|S_\varphi| < k/\Delta$, then the chances are low.

We see that V has two main tasks in the protocol: picking $h \in \mathcal{H}$ and checking that the value of x returned by P indeed satisfies $x \in S_\varphi$ and $h(x) = 0^l$. Looking at the runtime, we have seen how to pick h in polynomial time, and we are given that V can check if $x \in S_\varphi$ in polynomial time. Evaluating $h(x)$ is also doable in polynomial time (but not necessarily its inverse).

Recall that by definition of $\mathcal{H}$, the following is true $\forall x \neq y \in \{0,1\}^n, \forall a, b, \in \{0,1\}^l$:

$$Pr_{h \in \mathcal{H}}[h(x) = a \wedge h(y) = b] = 2^{-2l}.$$

We will now prove that $|h(S_\varphi)| \approx |S_\varphi|$ with the following lemma.

Lemma 6. *Given $\mathcal{H}$ as above and set $U \subseteq \{0,1\}^n$ with $a = |U|/2^l$, then*

$$Pr[0^l \in h(U)] \in [a - \frac{a^2}{2}, a].$$

Proof. Upper bound: $\forall x, Pr_{h\in\mathcal{H}}[0^l = h(x)] = 2^{-l}$, so by union bound

$$Pr_{h\in\mathcal{H}}[0^l \in h(U)] \leq \sum_{x\in U} Pr_{h\in\mathcal{H}}[0^l = h(x)] = |U|/2^l = a.$$

Lower bound: We note that the Principle of Inclusion and Exclusion holds here, and that the first two terms will provide a lower bound:

$$\begin{aligned} Pr_{h\in\mathcal{H}}[0^l \in h(U)] &\geq \sum_{x\in U} Pr_{h\in\mathcal{H}}[0^l = h(x)] - \sum_{x\neq y\in U} Pr_{h\in\mathcal{H}}[0^l = h(x) = h(y)] \\ &= \sum_{x\in U} 2^{-l} - \sum_{x\neq y\in U} 2^{-2l} = a - \binom{|U|}{2} 2^{-2l} \geq a - \frac{a^2}{2}. \end{aligned}$$

□

All that remains is to show that our protocol satisfies Claim 5. We let $U = S_\varphi$ so $a = |S_\varphi|/2^l$. Then, we see that:

- if $|S_\varphi| > k$, then $a \geq 1/2$, so by Lemma 6, $Pr[0^l \in h(S_\varphi)] \geq a - a^2/2 = 3/8$;
- if $|S_\varphi| < k/4$, then $a < \frac{k/4}{2^l} \leq 1/4$, so by Lemma 6, $Pr[0^l \in h(S_\varphi)] \leq 1/4$.

This means we can just repeat the protocol many times, each time with a random $h \in \mathcal{H}$, and then apply Chernoff bounds to get the desired probabilistic bounds.

Note: all randomness in this protocol was public. This leads to the following idea: if P can show that a lot of private random strings cause V to accept, then this can convince V that using public random strings is sufficient. More to come...