6.889 Sublinear Time Algorithms	March 11, 2019

Lecture 10

Lecturer: Ronitt Rubinfeld — *Scribe: Rogers Epstein*

1 Lower Bounds via Yao's Principle

This approach actually works for any randomized algorithm! But we will use it for sublinear ones. Overall, this is a game theoretic viewpoint introduced by Andrew Yao in 1977 to lower bound the query complexity of a randomized algorithm.

Principle 1 *If $\exists$ probability distribution D on a union of "positive" and "negative" inputs (on which the algorithm should output "yes" or "no," respectively) such that any deterministic algorithm of query complexity $\leq t$ is incorrect with probability $\geq \frac{1}{3}$, then t is a lower bound on the randomized query complexity of this language.*

Moral of Yao's: Average case lower bound on a deterministic algorithm implies a worst case lower bound on randomized algorithms.

Note that D must be over a large $(\omega(t)))$ number of inputs, otherwise there always exists an easy algorithm that outputs the right answer and fails the conditional statement.

For the game theoretic interpretation, suppose we have a game where Alice gets to choose any deterministic algorithm A, and Bob chooses some input x. Here, the range of A is always $\{0, 1\}$. Here the outcome of the game is whether or not A gives the right answer (which we represent by a cost of 0) or the wrong answer (which we represent by a cost of 1). In this case, Alice aims to minimize her cost and Bob aims to maximize it. Thus, the payoff matrix can be viewed as zero-sum, and we can apply the minimax theorem. This theorem implies that Bob has a randomized strategy that is at least as good when Alice's selection of A is randomized. This phrasing is analagous to the original statement of the principle.

2 Palindrome Example

Let $L = \{ww^R : w \in \{0, 1\}^{n/2}\}$. Here, w^R means the reverse of the string w. Thus, L is the set of n bit palindromes.

Definition 2 *w is ϵ-close to L if $\exists w' \in L$ such that w and w' are ϵ-close in Hamming distance (i.e. they differ in at most ϵn bits).*

We can easily check if some w is ϵ-close to L by checking if the bits at index i and $n - i$ match, and if there are more than ϵn mismatches, we fail the input. Thus, we can randomly sample $O(\frac{1}{\epsilon})$ such bits to create a sublinear algorithm with $o(1)$ probability of failure.

Now, let $L_n = \{vv^R ww^R : 2(|v| + |w|) = n\}$ be the set of n bit strings that are concatenations of two palindromes. We use the same definition of ϵ-close for some string to L_n.

Theorem 3 *To test if an input string is ϵ-close to L_n with error less than $\frac{1}{3}$, we need $\Omega(\sqrt{n})$ queries.*

We will prove this by finding a "bad" input distribution D for *any* deterministic algorithm. Then, by Yao, we get our randomized lower bound. For this construction, we assume $6|n$. Then:

- Our distribution on negative (ϵ-far) inputs will be the uniform distribution. Call this N.
- Our distribution on positive (in L_n) inputs will be constructed by first choosing a length $k = |v|$. Then we output a randomly constructed element of L_n with this constraint. Call this distribution P:

1. Randomly pick $k \in_R [\frac{n}{6}+1, \frac{n}{3}]$
2. Pick random v, u such that $|v| = k$ and $|u| = \frac{n-2k}{2}$.
3. Output $vv^R uu^R$ (which by construction is of length n and thus in L_n)

- Finally, our overall distribution is to sample from either N or P with equal probability of $\frac{1}{2}$.

Now we must show this distribution is "hard" for any deterministic algorithm using $\leq t = o(\sqrt{n})$ queries. We can model any such algorithm as a binary decision tree of depth t, where children of a node represent the next index queried given the information from querying all nodes from the root down to it. WLOG, all leaves reach depth t, where leaves represent the output of the algorithm, either P or N ("yes" or "no). Note there are $2^t = 2^{o(\sqrt{n})}$ leaves.

3 Error of a Leaf

Let $E^-(l) = \{$inputs $w \in \{0,1\}^n : w$ is ϵ-far from L_n and w reaches leaf $l\}$. Similarly, let $E^+(l) = \{$ inputs $w \in \{0,1\}^n : w \in L_n$ and w reaches leaf $l\}$. These are the inputs that reach a leaf that the algorithm should fail or pass, respectively. We aim to show that these sets have equal size at every leaf.

For notation, let PL be the set of leaves which are passing, and PF be the set of leaves that are failing. The total error of A on D is then

$$\sum_{l \in PL} Pr_{w \in D}[w \in E^-(l)] + \sum_{l \in PF} Pr_{w \in D}[w \in E^+(l)] \tag{1}$$

To analyze this quantity, we present two claims. They essentially say that each leaf node sees about the number of both "negative" and "positive" inputs you'd expect (since we expect about 2^{-t} fraction of either to end up at each leaf.

Claim 4 *If $t = o(n)$, $\forall l$ at depth t, $Pr_D[w \in E^-(l)] \geq (\frac{1}{2} - o(1)) * 2^{-t}$.*

Claim 5 *If $t = o(\sqrt{n})$, $\forall l$ at depth t, $Pr_D[w \in E^+(l)] \geq (\frac{1}{2} - o(1)) * 2^{-t}$.*

Under these claims, the error can be simplified from Equation 1:

$$\sum_{l \in PL} Pr_{w \in D}[w \in E^-(l)] + \sum_{l \in PF} Pr_{w \in D}[w \in E^+(l)]$$
$$\geq \sum_{l \in PL} (\frac{1}{2} - o(1)) * 2^{-t} + \sum_{l \in PF} (\frac{1}{2} - o(1)) * 2^{-t} = \frac{1}{2} - o(1) >> \frac{1}{3}$$

This gives us the desired result, so it only remains to prove these claims.

4 Proof of Claims

First, we prove Claim 4:

Proof If we pick $w \in_R \{0,1\}^n$, then the probability that w reaches l must be $\frac{1}{2^t}$ since for each of the t index queries, w is equally likely to have a 0 or 1.

Next, note that $|L_n| \leq \frac{n}{2} 2^{n/2}$ since there are $\frac{n}{2}$ possible lengths for v $(2, 4, 6, ..., n)$, and exactly $\frac{n}{2}$ degrees of freedom when assigning the bits in the first halves of v and w.

Additionally, the number of words that are ϵ-close from L_n is at most $|L_n| \sum_{i=0}^{\epsilon n} \binom{n}{i} \leq 2^{\frac{n}{2} + 2\epsilon n \log \frac{1}{\epsilon}}$, since we can construct all of them by choosing an element of L_n and changing at most ϵn bits. Here we've applied a Chernoff bound on the Binomial Distribution.

So, $|E^-(l)| \geq 2^{n-t} - 2^{\frac{n}{2}+2\epsilon n \log \frac{1}{\epsilon}}$. If $t = o(n)$ and we choose $\epsilon << \frac{1}{8}$ so that $n - t >> \frac{n}{2} + 2\epsilon n \log \frac{1}{\epsilon}$, we can lower bound the above expression by $(1 - o(1))2^{n-t}$.

Then, $Pr_D[w \in E^-(l)] = \frac{1}{2} Pr_N[w \in E^-(l)] \geq \frac{1}{2}\frac{|E^-(l)|}{2^n} \geq (\frac{1}{2} - o(1))2^{-t}$. ■

Now, we prove Claim 5:

Proof For leaf l, let $Q_l = \{q_1, q_2, ..., q_t\}$ be the indices queried on the path to l. Since there are $\binom{t}{2}$ pairs of q_i, q_j in Q, then for each pair there are at most 2 choices (possibly 1) of k such that q_1, q_2 symmetric with respect to k or $\frac{n-2k}{2}$.

Thus, the number of choices of "good k" such that no pair in Q_l are symmetric around k or $\frac{n-2k}{2}$ is at least $\frac{n}{6} - 2\binom{t}{2} = (1 - o(1))\frac{n}{6}$ since $t = o(\sqrt{n})$. For these "good k," the number of strings that follow this path is $2^{\frac{n}{2}-t}$.

So, $Pr_P[w \in E^+(l)] = \sum_w Pr_P[w] 1_{w \in E^+(l)} = \sum_w \sum_l Pr_P[w|k] Pr[k] 1_{w \in E^+(l)}$

$$= \sum_k \sum_w 2^{-\frac{n}{2}} \frac{6}{n} 1_{w \in E^+(l)} \geq \frac{6}{n} 2^{-\frac{n}{2}} * (1 - o(1)) \frac{n}{6} * 2^{\frac{n}{2}-t} = (1 - o(1))2^{-t}$$

Finally, we then get $Pr_D[w \in E^+(l)] = \frac{1}{2} Pr_P[w \in E^+(l)] \geq (\frac{1}{2} - o(1))2^{-t}$. ■

Thus, we've proven Theorem 3 by applying Yao.