Lecture 6: Yao's min-max for randomized query complexity

Rajat Mittal

IIT Kanpur

I would like to thank Vatsal Jha for making a preliminary version of these lecture notes.

1 Randomized query complexity

Recall that a randomized decision tree, say A, computing a boolean function f is a probability distribution over the Deterministic Decision trees.

Formally, we say that A computes f with bounded error ϵ if for every x we have:

$$\Pr_{D\sim A}(D(x) = f(x)) \geq 1 - \epsilon,$$

Here the probability is taken over all deterministic trees D having non-zero probability in the distribution defined by A. For simplicity, we will use A to represent both the randomized decision tree as well as the distribution defined by it.

The complexity (or cost) of a randomized decision tree A for an input x, denoted as $cost(A, x)$, has at least two possible definitions. Note that the cost on x for a deterministic tree D is the number of queries by D on x.

- Expected cost of computing x, where the expectation is taken according to the probability distribution represented by A,
$$cost(A, x) := \mathop{\mathrm{E}}_{D\sim A}[cost(D, x)].$$
- The worst-case cost, i.e.,
$$cost(A, x) := \max_{D\in A:\Pr(D)\neq 0} cost(D, x).$$

Given $cost(A, x)$, similar to the Deterministic tree complexity, the randomized decision tree complexity of a function f with bounded error ϵ is defined as:

$$\mathrm{R}_\epsilon(f) := \min_{A:A \text{ computes } f} \max_x cost(A, x).$$

We will see that for the bounded error case the above two definitions of $cost(A, x)$ are equivalent, in the sense that they can differ by at most a constant factor. For brevity of notation, $\mathrm{R}^1_\epsilon(f)$ will denote the worst-case cost while $\mathrm{R}^2_\epsilon(f)$ will denote the expected cost.

We now relate the two measures for randomized costs.

Theorem 1. $\mathrm{R}^1_\epsilon(f) = \theta(\mathrm{R}^2_\epsilon(f))$.

Proof. From the definition, $\mathrm{R}^2_\epsilon(f) = O(\mathrm{R}^1_\epsilon(f))$. We will show $\mathrm{R}^1_\epsilon(f) = O(\mathrm{R}^2_\epsilon(f))$ to finish the proof.

For simplicity let us assume $\epsilon = 1/3$. For a randomized decision tree A, let $d := \mathrm{R}^2_\epsilon(f)$.

We now devise a randomized algorithm $A^{'}$, using A, such that the *worst-case* cost of $A^{'}$ is of the same order as the expected cost of A. Run A up to $\geq 100d$ queries, if we get an answer then output it, otherwise answer arbitrarily.

Now we show below that $A^{'}$ computes f with high probability. Remember that the error probability can be made small by repetition. By Markov's inequality, the probability that a deterministic tree has depth more than $100d$ is less than $1/100$.

Exercise 1. What can we say about the error probability of $A^{'}$?

We leave the rest of the proof as an assignment. □

Generally, the complexity of a randomized decision tree A is taken to be its worst-case cost. Following the same convention henceforth, by $\mathrm{R}_\epsilon(f)$ we would refer to $\mathrm{R}^1_\epsilon(f)$. Also, for simplicity ϵ will be fixed to $\frac{1}{3}$. In fact any $\epsilon < \frac{1}{2}$ would have sufficed because of the Chernoff bound.

Note 1. It is sufficient that the error probability here is less than $1/2 - \epsilon$ for a constant ϵ. It is not enough to show that the error probability is $< 1/2$.

Randomized query complexity for OR: We now consider the randomized query complexity of OR. You will show in the assignment that there is a $2n/3$ query algorithm which computes OR in bounded error setting. Approximate degree gives a lower bound of $\sqrt{n}$ on OR. Can we make a better algorithm (asymptotically) for OR?

It can be shown that the approximate degree of OR is $\sqrt{n}$, so there is no hope of getting a better lower bound using the previous technique. We will show an extension of adversary technique for randomized query complexity; using this technique, we will show that indeed the trivial algorithm for OR is indeed the best ($\mathrm{R}_{1/3}(\mathrm{OR}) = \Omega(n)$).

2 Another lower bound technique for randomized query complexity

We will see a general extension (works for any function and not just OR) of the adversary method. Unlike the deterministic case, adversary will define a *hard* distribution on inputs such that no deterministic tree of small depth (say $\leq d$) will have good probability of success against this distribution of inputs.

Yao's min-max proves that this is enough to show that $R_{1/3}(f)$ is high (more than d). To go through the details, let us learn some definitions.

2.1 Yao's Minimax Lemma

Yao's Minimax lemma provides a general strategy for lower bounding randomized algorithms. It relates the distributional complexity for a boolean function f to $\mathrm{R}_\epsilon(f)$.

Distributional Complexity: The distributional complexity of a boolean function f with respect to a distribution μ on the inputs $\{0,1\}^n$ is defined as:

$$D_\mu(f) := \min_{D:\ D \text{ computes } f \text{ on } \mu} \max_x cost(D, x),$$

where a deterministic decision tree D is said to compute f on μ if $\Pr_{x\sim\mu}(D(x) = f(x)) \geq \frac{2}{3}$.

Lemma 1. *(Yao's Minimax Lemma) Let f be a boolean function and let μ be a distribution over the inputs. Then Yao's Minimax Lemma states that:*

$$\max_\mu D_\mu(f) = \mathrm{R}_{1/3}(f).$$

Proof. We first show that $\max_\mu D_\mu(f) \leq \mathrm{R}_{1/3}(f)$.

Let A be an optimal randomized decision tree computing f i.e. $cost(A) = \mathrm{R}_{1/3}(f)$ with the associated distribution to be λ. Further, consider a distribution μ over the inputs $\{0,1\}^n$.

Our aim will be to show that there exists a decision tree D with non-zero probability in the support of λ such that D computes f according to μ. The argument used will be the probabilistic version of equating the sum of entries of a matrix using columns and rows.

Let $1_{(x,D)}$ denote the indicator function that $D(x) = f(x)$. This implies for all x we have:

$$\mathop{\mathrm{E}}_{D\sim\lambda}[1_{(x,D)}] = \sum_{D\sim\lambda} 1_{(x,D)} \Pr(D) = \sum_{D:D\sim\lambda,\ D(x)=f(x)} \Pr(D) \geq \frac{2}{3}.$$

Now, consider the following expression:

$$\mathop{\mathrm{E}}_{x\sim\mu,\ D\sim\lambda}[1_{(x,D)}] = \sum_{x\sim\mu,\ D\sim\lambda} 1_{(x,D)} \Pr(x, D).$$

As picking x and picking D are independent of each other we obtain:

$$\begin{aligned}\mathop{\mathrm{E}}_{x\sim\mu,\ D\sim\lambda}[1_{(x,D)}] &= \sum_{x\sim\mu,\ D\sim\lambda} 1_{(x,D)} \Pr(x) \Pr(D) \\ &= \sum_{x\sim\mu} \Pr(x) \sum_{D\sim\lambda} 1_{(x,D)} \Pr(D)\end{aligned}$$

Using $\sum\limits_{D\sim\lambda} 1_{(x,D)} \Pr(D) \geq \frac{2}{3}$, we obtain

$$\mathop{\mathrm{E}}_{x\sim\mu,\ D\sim\lambda}[1_{(x,D)}] \geq \frac{2}{3} \sum_{x\sim\mu} \Pr(x) = \frac{2}{3}. \tag{1}$$

Now taking the inner sum over x instead of D in the above double summation gives us:

$$\mathop{\mathrm{E}}_{x\sim\mu,\ D\sim\lambda}[1_{(x,D)}] = \sum_{D\sim\lambda} \Pr(D) \sum_{x\sim\mu} \Pr(x) 1_{(x,D)} = \sum_{D\sim\lambda} \Pr(D) \sum_{x\sim\mu,\ D(x)=f(x)} \Pr(x).$$

Since the expectation of $\sum_{x\sim\mu,\ D(x)=f(x)} \Pr(x)$ with respect to λ is bigger than $2/3$, there exist a D in the support of λ satisfying:

$$\mathop{\mathrm{E}}_{x\sim\mu}[1_{(x,D)}] = \sum_{x\sim\mu,\ D(x)=f(x)} \Pr(x) \geq \frac{2}{3}.$$

The above claim is equivalent to claiming that there exists a D in λ which computes f according to the distribution μ. As the choice of μ was arbitrary this implies:

$$\max_{\mu} D_{\mu}(f) \leq \mathrm{R}_{1/3}(f).$$

Now to prove the equality, it suffices to show that there exists a distribution μ over the inputs such that $D_{\mu}(f) = \mathrm{R}_{\mu}(f)$.

The existence of such a μ follows from duality theorems in linear programming. This completes the proof of Yao's Minimax lemma. □

Having proved the Yao's Minimax lemma we now proceed to obtain a lower bound on $\mathrm{R}_{1/3}(\mathrm{OR})$ using the adversary argument.

2.2 Lower bound for $\mathbf{R_{1/3}(OR)}$

We are ready to prove that the randomized query complexity of OR is $\Theta(n)$.

Theorem 2.

$$\mathrm{R}_{1/3}(\mathrm{OR}) \geq \frac{n}{2}.$$

Proof. By Yao's min-max lemma, it is enough to prove that $\max_{\mu} D_\mu(\text{OR}) \geq \frac{n}{2}$. In other words, we need to show a μ such that $D_\mu(\text{OR}) \geq \frac{n}{2}$. Define μ, a probability distribution on the inputs as

$$\Pr(X = x) = \begin{cases} \frac{1}{3} + \frac{1}{n}, & \text{if } |x| = 0 \\ \\ \frac{2}{3n} - \frac{1}{n^2}, & \text{if } |x| = 1 \end{cases}.$$

Exercise 2. Verify that it is a probability distribution as required in Yao's min-max lemma.

Now consider any decision tree D having depth $< \frac{n}{2}$ which computes OR according to the distribution μ. Look at the branch of D where we get 0 in all the queries. Since the probability of 0 input is $\frac{1}{3} + \frac{1}{n}$ and D computes OR on μ with error probability less than $1/3$, D should output 0 on this *all zero* branch.

In other words, for the all zero branch of D if the output on that branch is 1 then the error probability of D is $\frac{1}{3} + \frac{1}{n} > 1/3$, a contradiction.

On the other hand, if the output is 0 on this all zero branch, then for more than half the inputs x having $|x| = 1$ the output will be 0 (remember that D has only queried less than $n/2$ indices till now). The error probability of D in this case will be at least :

$$\left(\frac{n}{2} + 1\right)\left(\frac{2}{3n} - \frac{1}{n^2}\right) = \frac{1}{3} + \frac{1}{6n} - \frac{1}{n^2} > \frac{1}{3},$$

for $n > 6$.

This again contradicts the assumption that D computes OR according to the given distribution μ with probability more than $2/3$.

As our choice of D was arbitrary, by the above argument we have proved that there cannot exist a D with depth $< n/2$ that computes OR according to the distribution μ defined above. So $D_\mu(\text{OR}) \geq \frac{n}{2}$.

Hence,

$$\text{R}_{1/3}(\text{OR}) = \max_{\mu} D_\mu(\text{OR}) \geq \frac{n}{2}.$$

□

It can be observed that the key step in the adversary argument provided above was to identify the *hard* distribution μ, in order to apply Yao's lemma. In other words, Yao's lemma evidently provides an adversary strategy for lower bounding $\text{R}_\epsilon(f)$.

3 Assignment

Exercise 3. Read about duality theory of linear programming.

Exercise 4. Finish the proof of Theorem 1.

Exercise 5. Let f be the Majority function. Prove that $\text{R}_{1/3}(f) = \Omega(n)$.

References