Lecture (28^{th} Sept): Partial Functions

Scribe: Debarsho Sannyasi   Lecture: Rajat Mittal

IIT Kanpur

In the last lecture, we were introduced to the topic of an approximate polynomial of a Boolean function, and approximate degree can be used to find lower bounds of quantum query complexity ($Q_\epsilon(f)$). We have also seen a method to lower bound approximate degree of symmetric functions using symmetrization and forming a univariate polynomial, using which we derived that $\widetilde{deg}_{1/3}(PARITY) = n$ and $\widetilde{deg}_{1/3}(OR) = \Omega(\sqrt{n})$ ($\widetilde{deg}_{1/3}(OR)$ is indeed $\Theta(\sqrt{n})$ using the bound on number of queries in Grover search algorithm).

In this lecture, we will start a new topic called partial functions, and how their properties might differ from total functions.

1 Partial functions

Let $g : \{0,1\}^n \to \{0,1\}$ be a Boolean function. Its domain is the complete Boolean hypercube $\{0,1\}^n$. Such a function is called a total function. Most of the functions we have seen till now are total functions. But in many cases, we do not need to specify the function on the entire Boolean hypercube.

A function $f : D \to \{0,1\}$ where $D \subseteq \{0,1\}^n$ is the domain of f is called a partial Boolean function. Notice that, f needs to be specified only for $x \in D$. For all $x \in \{0,1\}^n/D$, we do not care about the value $f(x)$. Do not confuse into thinking that partial parities are partial functions. Partial parities are total functions as their domain is the entire Boolean hypercube $\{0,1\}^n$ and the function is specified for each $x \in \{0,1\}^n$. One major importance of partial functions is that they are analogs of *promise problems* in Complexity Theory. *Promise* comes from the fact that the input to these problems is promised to be from a subset of $\{0,1\}^*$. We do not care about the output for inputs outside this subset. Such problems are common/natural and thus useful in Approximation Theory and Complexity. Goldreich[1] is a nice survey on promise problems.

1.1 Some examples of partial functions

- **Promise-OR**: Defined on only those inputs which have Hamming weight either 0 or 1, that is the set $\{x \in \{0,1\}^n : |x| = 0 \text{ or } |x| = 1\}$. So there are $n+1$ inputs on which this function is specified.
  Recall that when we derived $R_{1/3}(OR) = \Omega(n)$, the *hard* distribution only included inputs having hamming weights 0 or 1. Hence, using similar arguments as OR, we can derive that D(Promise-OR)$= n$ and $R_{1/3}$(Promise-OR)$= \Omega(n)$.
- **Gap Majority:** Defined on only those inputs which have Hamming weight either $\frac{n}{2} + \sqrt{n}$ or $\frac{n}{2} - \sqrt{n}$. If $|x| = \frac{n}{2} + \sqrt{n}$, then $gapmaj_n(x) = -1$, and if $|x| = \frac{n}{2} - \sqrt{n}$, then $gapmaj_n(x) = 1$, where $gapmaj_n$ is the Gap Majority function.
  Recall that the composition of R_ϵ, that is $R_\epsilon(f \circ g) = R_\epsilon(f) \cdot R_\epsilon(g)$ is still an open problem. It was recently proven that if $R_\epsilon(f \circ gapmaj_n) = R_\epsilon(f) \cdot R_\epsilon(gapmaj_n)$ is true, then $R_\epsilon(f \circ g) = R_\epsilon(f) \cdot R_\epsilon(g)$ is also true for any other inner function g [2]. Thus, we need to only examine the case when the inner function is $gapmaj_n$ to conclude whether the outer function f will compose or not. Indeed for any symmetric function f, $R_\epsilon(f \circ gapmaj_n) = R_\epsilon(f) \cdot R_\epsilon(gapmaj_n)$ holds true.
- Graph properties on special kinds of graphs like planar graphs, connected graphs, etc.: Recall that any graph with n vertices can be specified using a binary string of length $\binom{n}{2}$. So, the set of representations of special kinds of graphs such as planar or connected graphs is a subset of $\{0,1\}^{\binom{n}{2}}$. Thus, the input to the algorithm is promised to be belonging to such subsets. The graph property or the algorithm can be seen as partial functions.

We have seen the importance and usefulness of partial functions, but we need to be *careful* while dealing with partial functions. This is because it may happen that some properties or results are true for total functions, but are not true for partial functions. So, be careful while using such results or making arguments for partial functions. In the rest of the lecture, we will see such cases.

1.2 Approximate degree of Promise-OR

We have already seen that D(Promise-OR)$= n$ and $R_{1/3}$(Promise-OR)$= \Omega(n)$ which are no better than that for OR, but what can we say about $\widetilde{deg}_{1/3}$(Promise-OR), is it lower than $\widetilde{deg}_{1/3}$(OR)? Let f be a partial function defined on domain $D \subseteq \{0,1\}^n$. Let the approximating polynomial of f be p. One obvious condition which must satisfy is that $|p(x) - f(x)| \leq 1/3 \ \forall \ x \in D$. But we should also specify the behaviour of p on any $x \in \{0,1\}^n/D$. For this, we have the following two cases:

1. Unbounded approximate degree, $\widetilde{deg}^{u}_{1/3}(f)$: $p(x)$ can be any value $\forall \ x \in \{0,1\}^n/D$.
2. Bounded approximate degree, $\widetilde{deg}^{b}_{1/3}(f)$: $-1/3 \leq p(x) \leq 4/3 \ \forall \ x \in \{0,1\}^n/D$.

Clearly, $\widetilde{deg}^{u}_{1/3}(f) \leq \widetilde{deg}^{b}_{1/3}(f)$ as there are more constraints in the bounded case. So, the bounded case will be better if we want to lower bound some quantity, such as $Q_{1/3}(f)$ as done in the last lecture. Also, we are talking about approximate polynomials for Boolean functions, it makes more sense in some applications if $p(x)$ is at least bounded for non-domain inputs. It can have any value on non-domain inputs but inside a bounded range. Recall, while lower bounding quantum query complexity, the probability of acceptance was a polynomial and as its a probability measure, the range must be $[0,1]$ for all inputs.

Now, for Promise-OR, the unbounded approximate degree is 1. Consider the polynomial $p(x) = x_1 + x_2 + \cdots + x_n$ whose degree is 1. We can easily see that on all inputs x such that $|x| = 0$ or $|x| = 1$, $p(x)$ is indeed equal to Promise-OR(x). Thus, $\widetilde{deg}^{u}_{0}$(Promise-OR)=1.

Now we will show that $\widetilde{deg}^{b}_{1/3}$(Promise-OR)=$\Omega(\sqrt{n})$, which is the same as that for OR. Recall that while proving $\widetilde{deg}_{1/3}(OR) = \Omega(\sqrt{n})$, we used only two things. Firstly, the approximate polynomial P should stay in between a narrow band, i.e. $P(w) \in [2/3, 4/3] \ \forall \ w \in \{1, 2, \ldots, n\}$ (w is the hamming weight of the input and P is the approximate polynomial for OR) and $P(0) \in [-1/3, 1/3]$. A similar thing must hold true for the bounded case as well. The approximate polynomial G of Promise-OR should stay in between a narrow band $[-1/3, 4/3] \ \forall \ w$. Secondly, there should be a sharp derivative. In the case of OR, the share derivative was in between $w = 0$ and $w = 1$, the same thing is true for the bounded case as well as the domain includes inputs with $w = 0$ and $w = 1$. Any polynomial satisfying these conditions must have degree $\Omega(\sqrt{n})$ as seen in the last lecture. Thus, we get $\widetilde{deg}^{b}_{1/3}$(Promise-OR)=$\Omega(\sqrt{n})$.

Indeed one can see that similar argument hold for any non-constant total symmetric function f. Clearly the range of the approximating polynomial needs to be bounded, and as f is non-constant, there exists some hamming weights t and $t+1$ where the function changes its value from 0 to 1 or vice versa providing the sharp derivative. Thus, the approximate degree of any non-constant symmetric total function is at least $\Omega(\sqrt{n})$, that is $\widetilde{deg}_{\epsilon}(f) \geq \Omega(\sqrt{n})$.

Note: As t goes closer and closer to $n/2$, the approximate degree keeps rising. Specifically for MAJ_n, t is closest to $n/2$ ($t = n/2$ for even n) and its approximating degree is $\Theta(n)$.

So we saw how properties of partial functions might differ for the approximate degree. Now, let us study a theorem that is true for total functions but not true for partial functions.

2 $R_\epsilon(f)$ of non-constant symmetric functions

Query complexity of any constant function is 0. So let us focus on non-constant symmetric functions. We have seen that $R_\epsilon(OR) = \Omega(n)$. We will prove in this section that $R_\epsilon(f) = \Omega(n)$ for any symmetric non-constant total function f. In this section, we will talk about total functions unless mentioned otherwise.

Note: $D(f) = n$ for all symmetric non-constant function f. As f is not constant, without loss of generality there exists some $t < n$ for which $f(x) = 0$ ($\forall\ x : |x| = t$) and $f(x) = 1$ ($\forall\ x : |x| = t+1$) (or equivalently vice versa). Now the adversary argument answers the first t queries as 1, and all the further queries until and including $n-1^{th}$ query as 0. Thus, the final output value depends upon the n^{th} queried bit.

Let f be a non constant symmetric function. So there exist some t such that $f(x) = b$ ($\forall\ x : |x| = t$) and $f(x) = 1-b$ ($\forall\ x : |x| = t-1$). Without loss of generality we can assume that $\exists t \leq \frac{n}{2}$ such that $f(x) = 1$ ($\forall\ x : |x| = t$) and $f(x) = 0$ ($\forall\ x : |x| = t-1$). We can assume this because if the function value is 0 at a higher Hamming weight (that is t), and 1 at a lower Hamming weight (that is $t-1$), then we can simply consider the complement of that function as $g = 1-f$. And if such a $t > n/2$, we can simply relabel 0 as 1 and 1 as 0. Thus we have

$$\begin{aligned} \exists t \leq \frac{n}{2} : & f(x) = 1 \qquad \forall x : |x| = t \\ & f(x) = 0 \qquad \forall x : |x| = t-1 \end{aligned}$$

and we want to prove that $R_\epsilon(f) = \Omega(n)$. As for the case OR, we will use Yao's Minimax Lemma by constructing a *hard* distribution. Looking at the above condition, we can intuitively identify that the *hard* distribution μ on the inputs x will be

$$\begin{aligned} \mu(x) &= \frac{1}{2 \cdot \binom{n}{t}} = p_1 \qquad \forall x : |x| = t \\ \mu(x) &= \frac{1}{2 \cdot \binom{n}{t-1}} = p_2 \qquad \forall x : |x| = t-1 \\ \mu(x) &= 0 \qquad \textit{for all other } x \end{aligned}$$

Following Yao's Minimax Lemma, we want to show that for any deterministic tree D of depth at most cn (for some constant c)

$$Pr_{x\sim\mu}[D(x) \neq f(x)] > \epsilon$$

for some constant ϵ. From which we can conclude that $R_\epsilon(f) > cn \implies R_\epsilon(f) = \Omega(n)$. For convenience let us consider a deterministic tree D of depth $n/100$ (that is $c = 1/100$). Now equivalently we can also consider that D is a complete binary tree of depth $n/100$. This is because we can extend any leaf l outputing b at depth less than $n/100$ to a depth equal to $n/100$ by simply answering b at any leaf in the subtree rooted at l. So every path in D from root to any leaf will be of length $n/100$.
We will now have the following three cases:

1. **Case 1:** $t \geq n/50$
   The depth of D is $n/100$. So for any path from the root to any leaf of the tree, we can see at most $n/100$ 1's. Now, this path can be extended to inputs with Hamming weight t as well as to inputs with Hamming weight $t-1$. Let l be a leaf of the tree. Let set A_l be the set of inputs x which have taken the path from the root to leaf l and has $|x| = t$. Let set B_l be the set of inputs x which have taken the path from the root to leaf l and has $|x| = t-1$.
   The leaf l can answer correctly to inputs either in set A_l or to the inputs in set B_l.
   Let the number of 1's seen on the path from root to leaf l be y, and l is at depth $n/100$. So for any $x \in A_l$, x has the remaining $t-y$ 1's in the rest $99n/100$ bits. So

$$|A_l| = \binom{\frac{99n}{100}}{t-y}$$

and similarly

$$|A_l| = \binom{\frac{99n}{100}}{t-1-y}$$

Each input in A_l has probability p_1 and each input in B_l has probability p_2. The error probability for inputs reaching leaf l will be

$$\epsilon = min\left(\frac{p_1|A_l|}{p_1|A_l| + p_2|B_l|}, \frac{p_2|B_l|}{p_1|A_l| + p_2|B_l|}\right)$$

The above quantity is a constant. See Appendix for exact calculations. So, for any leaf, we are computing the answer incorrectly for a constant fraction of inputs, which implies the tree D is computing an incorrect answer for a constant fraction of inputs.

2. **Case 2:** $t \leq 100$
Here we will have a similar argument to that of when we proved the same for OR. Now consider the path from the root to leaf l which has all 0's. l is at depth $n/100$ as D is a complete binary tree. Let set A_l be the set of inputs x which have taken this path from the root to leaf l and has $|x| = t$. Let set B_l be the set of inputs x which have taken this path from the root to leaf l and has $|x| = t - 1$.
If l answers 1, then it would answer incorrectly for all inputs in B_l, and if l answers 0, then it would answer incorrectly for all inputs in A_l. Thus, the **overall** error will at least be

$$\epsilon = min\left(p_1|A_l|, p_2|B_l|\right)$$

$|A_l| = \binom{99n/100}{t}$ and $|B_l| = \binom{99n/100}{t-1}$. We can easily conclude that ϵ will be a constant if $t \leq 100$. See Appendix for exact calculations.

3. **Case 3:** $100 < t < n/50$
This is a tricky case. We will only give an intuitive argument for this case. D is a complete binary tree of depth $n/100$. Now let say $t = n/100$. The optimal tree will assign 1 at a leaf which we reach from the root by following all 1's. So at this leaf, all the inputs are answered correctly. For the leaves to which we arrive seeing most of the inputs as 1, we simply consider that D is answering correctly on all inputs reaching such leaves. But for all other leaves, D is answering incorrectly for a constant fraction of inputs.

In other words, let for inputs following the path in which we have the number of 1's between $7t/8$ and t be answered correctly by D. The fraction of such inputs will be very small. The number of such paths will be $\binom{n/100}{7t/8} + \binom{n/100}{7t/8+1} + \cdots + \binom{n/100}{t}$ and corresponding weight will be small. Note that we can choose a much higher constant than $7/8$ such as 0.9999 which will make this quantity even smaller.

Now, look at all the paths from the root to any leaf where the number of 1's is less than $7t/8$. Most of the inputs will lie on such paths. The total number of such paths will be $\binom{n/100}{0} + \binom{n/100}{1} + \cdots + \binom{n/100}{7t/8-1}$. Note that we can further increase the constant from $7/8$ to 0.9999 to further strengthen this claim. On those leaves to which we reach following such paths will have a constant probability of answering incorrectly. This can be shown by a similar argument as Case 1 above. Let l be such a leaf that has seen $y < 7t/8$ 1's. We follow the same procedure as Case 1 and form sets A_l, B_l. l can only answer correctly to inputs belonging to one of the two sets which will give rise to incorrect answers on a constant fraction of inputs at this leaf. Since such leaves form the bulk of the inputs, overall error will also be a constant due to answering incorrectly for a constant fraction of inputs. In a way, Case 3 can be seen as a mixture of Cases 1 and 2, where we assume that D will answer correctly on paths with a high number of 1's but they will constitute a lower fraction of weights. The other paths will form the bulk of the inputs and will answer incorrectly for a constant fraction of inputs.

So any deterministic tree D of depth $n/100$ is answering incorrectly to a constant fraction of all inputs according to the distribution μ. Thus, finally we have proven that $R_\epsilon(f) = \Omega(n)$ for a non constant symmetric function f. Recall our first assumption about the existence of Hamming weight $t \leq n/2$ such that $f(x) = 1$ $(\forall\ x : |x| = t)$ and $f(x) = 0$ $(\forall\ x : |x| = t - 1)$. This is true only for total symmetric functions, and may not hold true for partial functions. Thus, the above proof may not hold true for a partial function. Take this example of a partial function g: $g(x) = 1\ \forall\ x : |x| \geq 2n/3$ and $g(x) = 0\ \forall\ x : |x| \leq n/3$. We can see that $R_\epsilon(g) = O(1)$. Toss a coin a constant number of times, if heads come up more than half the number of times, we answer 1, and if tails come up more than half the number of times, we answer 0. Each toss of the coin is analogous to querying a bit. If the number of heads is more than half the total number of tosses, then the probability of $|x| \geq 2n/3$ is greater than the probability of $|x| \leq n/3$. If the number of heads is less than half the total number of tosses, then the probability of $|x| \geq 2n/3$ is less than the probability of $|x| \leq n/3$, and thus this randomized algorithm will succeed with high probability. So we observe a drastic change in R_ϵ when we moved from total functions to partial functions.

As stated above, the existence of such a t for total functions is essential for proving the above theorem. For total functions, if $f(x) \neq f(y)$ for some x, y, then the path connecting x and y in the Boolean hypercube must contain two points z and w such that $f(z) \neq f(w)$ and their Hamming distance is 1. But this need not be true for partial functions and thus many proofs break down for partial functions. This is the missing link.

3 Appendix

In Section 2, for **Case 1**: $t \geq n/50$, we have

$$\epsilon = min\left(\frac{p_1|A_l|}{p_1|A_l| + p_2|B_l|}, \frac{p_2|B_l|}{p_1|A_l| + p_2|B_l|}\right)$$

Let us show that the first term inside the min() is $O(1)$ (that is a constant), a similar procedure will follow for the second term as well.

$$\begin{aligned}
\frac{p_1|A_l|}{p_1|A_l| + p_2|B_l|} &= \frac{\frac{1}{2\binom{n}{t}}\binom{99n/100}{t-y}}{\frac{1}{2\binom{n}{t}}\binom{99n/100}{t-y} + \frac{1}{2\binom{n}{t-1}}\binom{99n/100}{t-y-1}} \\
&= \frac{\frac{\frac{99n}{100}!}{\left(\frac{99n}{100}-(t-y)\right)!(t-y)!}}{\frac{\frac{99n}{100}!}{\left(\frac{99n}{100}-(t-y)\right)!(t-y)!} + \frac{\binom{n}{t}}{\binom{n}{t-1}}\frac{\frac{99n}{100}!}{\left(\frac{99n}{100}-(t-y)+1\right)!(t-y-1)!}} \\
&= \frac{\frac{1}{t-y}}{\frac{1}{t-y} + \frac{n-t+1}{t}\frac{1}{\frac{99n}{100}-(t-y)+1}} \\
&= \frac{1}{1 + \left(1 - \frac{y}{t}\right)\frac{n-t+1}{n-t+1+(y-\frac{n}{100})}} \\
&= \frac{1}{1 + \frac{1-(y/t)}{1+\frac{y-\frac{n}{100}}{n-t+1}}}
\end{aligned}$$

Now we have $n/2 \geq t \geq n/50$ and $y \leq n/100$, and thus the above quantity will be a constant.

For **Case 2**: $t \leq 100$, we have

$$\epsilon = min\left(p_1|A_l|, p_2|B_l|\right)$$

Lets compute $p_1|A_l|$ and show that its $O(1)$ ($p_2|B_l|$ would be similarly $O(1)$)

$$\begin{aligned} p_1|A_l| &= \frac{\binom{99n/100}{t}}{2\binom{n}{t}} \\ &= \frac{\frac{99n}{100}!(n-t)!t!}{2\cdot n!t!(\frac{99n}{100}-t)!} \\ &= \frac{1}{2}\frac{(99n/100)(99n/100-1)\cdots(99n/100-t+1)}{(n)(n-1)\ldots(n-t+1)} \end{aligned}$$

For large n and $t \leq 100$, the above quantity is $O(1)$.

References

1. O. Goldreich, "On promise problems: A survey," in *Theoretical computer science*. Springer, 2006, pp. 254–290.
2. S. Ben-David and E. Blais, "A tight composition theorem for the randomized query complexity of partial functions," in *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2020, pp. 240–246.