COMS 4995: Introduction to Communication Complexity — Sep 25, 2013

Lecture 4 – Randomized Decision Trees

Instructor: *Xi Chen* — Scribes: *Clément Canonne*

1 General Evasiveness (continued)

Recall that during the last lecture, we introduced the *Evasiveness Conjecture*, a generalization of the AKR conjecture. We also proved a specific case, when n is a prime power:

Theorem 1 (Rivest–Vuillemin)**.** *If $f\colon \{0,1\}^n \to \{0,1\}$ (where $n = p^k$ for some prime p) is invariant under a transitive permutation group and satisfies $f(\vec{0}) \neq f(\vec{1})$, then f is evasive.*

From this theorem, we show how one can derive the following corollary on monotone graph properties:

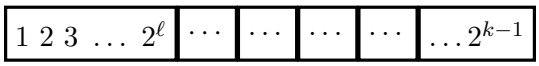
Corollary 2. *For any non-trivial monotone graph property $f\colon \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ with $n = 2^k$, we have $D(f) \geq n^2/4$.*

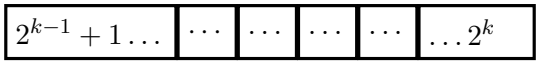
Proof. First, note that since $\binom{n}{2} = \frac{n(n-1)}{2} = \Theta(n^2)$, the result indeed does guarantee (weak) evasiness of the property. To prove it, we would like to apply Theorem 1; however, clearly $\binom{n}{2}$ is never of the form p^k.

Assume with have n vertices, where $n = 2^k$ for some $k \in \mathbb{N}$; we define a sequence of graphs $(G_j)_{0 \leq j \leq k}$, where:

- G_0 is the empty graph;
- G_j (for $1 \leq j \leq k$) is the disjoint union of 2^{k-j} complete graphs of size 2^j;

In particular, G_k is the complete graph on n vertices. Since by assumption f is non-trivial and monotone, $f(G_0) = 0$ and $f(G_k) = 1$; so that there must exist $\ell \in \{0, \dots, k-1\}$ for which $f(G_\ell) = 0$ and $f(G_{\ell+1}) = 1$. Fix such a ℓ, and partition the input variables in two blocks L, R of size 2^{k-1}, each of them itself divided in blocks of size 2^ℓ: this defines a "bipartite graph", where each subblock of 2^ℓ nodes is taken to be the complete graph K_{2^ℓ}, and two distinct subblocks in R (resp. L) share no edge. This graph contains between 0 and $2^{2k-2} = \frac{n^2}{4}$ *free* edges (that is, possible edges between L and R, as the edges within one given side are already fully specified by the construction).





Construct a Boolean function g on $n^2/4$ inputs as follows: for $x = (x_{i,j})_{(i,j) \in \{1,\dots,2^{k-1}\} \times \{2^{k-1}+1,\dots,2^k\}}$, $g(x) \stackrel{\text{def}}{=} f(G_x)$, where G_x is defined as the bipartite graph encoded in $(x_{i,j})$ augmented with the information

given out (that is, the structure of the graph described above). Clearly, $D(f) \geq D(g)$, and further $g(\vec{0}) = f(G_\ell)$, while $g(\vec{1}) = f(H) \geq f(G_{\ell+1})$, as H is a graph containing $G_{\ell+1}$ as a subgraph. It is easy to check that g also satisfies the other hypotheses of Rivest–Vuillemin; which, once applied with $p = 2$, yields the result.

□

2 Randomized Decision Trees

2.1 Definition

Definition 3 (Randomized Decision Tree)**.** *For $f\colon \{0,1\}^n \to \{0,1\}$, let $\mathcal{T}_f$ be the set of all deterministic (minimal)*[1] *decision trees that compute f. A* randomized decision tree for f *is a probability distribution p on $\mathcal{T}_f$. We denote by $p(T)$ the probability assigned by p to $T \in \mathcal{T}_f$.*

Definition 4 (Randomized Decision Tree Complexity)**.** *Given a randomized DT p for f and an input $x \in \{0,1\}^n$, the* expected number of queries *is $\mathbb{E}_{T\sim p}\,\mathrm{cost}(x,T)$. Accordingly, the* randomized decision tree complexity of f *is the quantity*

$$R(f) \stackrel{\text{def}}{=} \min_{p \text{ over } \mathcal{T}_f} \max_{x\in\{0,1\}^n} \mathbb{E}_{T\sim p}\,\mathrm{cost}(x,T) \tag{1}$$

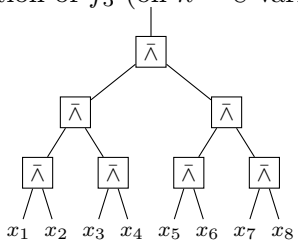
Example 5. Consider the function $\bar{\wedge}\colon (a,b) \to \neg(a \wedge b)$

$a \,\bar{\wedge}\, b$	a	b
1	0	0
1	1	0
1	0	1
0	1	1

and the function f_k on 2^k variables defined recursively by

$$\begin{aligned} f_0(x_1) &= x_1 \\ f_1(x_1x_2) &= f_0(x_1) \,\bar{\wedge}\, f_0(x_2) = x_1 \,\bar{\wedge}\, x_2 \\ f_k(x_1 \dots x_{2^{k-1}} x_{2^{k-1}+1} x_{2^k}) &= f_{k-1}(x_1 \dots x_{2^{k-1}}) \,\bar{\wedge}\, f_{k-1}(x_{2^{k-1}+1} x_{2^k}) \end{aligned}$$

For instance, here is the recursive definition of f_3 (on $n = 8$ variables):



[1] By *minimal*, we mean that no variable is queried more than once in any path from the root to the leaves – so that $|\mathcal{T}_f| < \infty$.

One can show that $D(f_k) = 2^k \stackrel{\text{def}}{=} n$; but what about $R(f)$?

A very simple randomized algorithm to evaluate f_k on x is to recursively go down the tree, picking a branch uniformly at random each time (randomized DFS) until enough information has been gathered to conclude with absolute certainty what $f_k(x)$ is. To analyze the expected number of queries this makes on an arbitrary $x \in \{0,1\}^n$, let

- a_k be the worst expected number of queries for f_k, over all inputs x satisfying $f_k(x) = 0$;
- b_k be the worst expected number of queries for f_k, over all inputs x satisfying $f_k(x) = 1$.

Clearly, $\begin{pmatrix} a_0 \\ b_0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$; and moreover

$$\begin{aligned} a_k &= 2b_{k-1} \\ b_k &= \frac{1}{2}a_{k-1} + \frac{1}{2}(b_{k-1} + a_{k-1}) = a_{k-1} + \frac{1}{2}b_{k-1} \end{aligned}$$

(to be certain that $f_k(x) = 0$, the algorithm has to make sure both f_{k-1}-subtrees evaluate to 1, and this gives us the $2b_{k-1}$; while to be certain that $f_k(x) = 1$, in the worst case one subtree evaluates to 0 and the other to 1, and the algorithm picks the "wrong" one (the latter) first and still has to be sure that the second one is 0 – which happens with probability half (giving the $\frac{1}{2}(b_{k-1} + a_{k-1})$; with probability half, it starts by looking at the 0-subtree, and thus simply has to be certain of this 0 outcome to conclude (hence the $\frac{1}{2}a_{k-1}$). We get the recurrence relation:

$$\begin{pmatrix} a_k \\ b_k \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & 2 \\ 1 & \frac{1}{2} \end{pmatrix}}_{M} \begin{pmatrix} a_{k-1} \\ b_{k-1} \end{pmatrix}$$

and one gets by solving the recurrence[2] that $a_k, b_k = \Theta\left(2^{0.754\ldots k}\right) = \Theta(n^{0.754\ldots})$

Observation 6. *This randomized decision tree complexity is actually optimal.*

2.2 Yao's Minimax Principle

We now turn to the description of a very important tool when dealing with randomized algorithms or protocols, *Yao's minimax principle* – also sometimes referred to as *von Neumann's minimax theorem*:

[2]To solve this, we diagonalize M (which has two distinct eigenvalues, and thus can indeed be diagonalized):

$$M = P \begin{pmatrix} \alpha & 0 \\ 0 & \beta \end{pmatrix} P^{-1}$$

with $P = \begin{pmatrix} -\beta & -\alpha \\ 1 & 1 \end{pmatrix}$, $\alpha = \frac{1-\sqrt{33}}{4}$ and $\beta = \frac{1+\sqrt{33}}{4}$. It follows that

$$\begin{pmatrix} a_k \\ b_k \end{pmatrix} = P \begin{pmatrix} \alpha^k & 0 \\ 0 & \beta^k \end{pmatrix} P^{-1} \begin{pmatrix} a_0 \\ b_0 \end{pmatrix} = \frac{1}{2\sqrt{33}} \begin{pmatrix} -2\sqrt{33}(\beta\alpha^k + \alpha\beta^k) \\ 5(\beta^k - \alpha^k) + \sqrt{33}(\alpha^k + \beta^k) \end{pmatrix} \underset{k\to\infty}{\sim} \beta^k \begin{pmatrix} \frac{8}{\beta} \\ \frac{5}{2\sqrt{33}} + \frac{1}{2} \end{pmatrix}$$

where the last step comes from observing that $\alpha\beta = -8$, and that $|\beta| > |\alpha|$ (so that $\left(\frac{\alpha}{\beta}\right)^k \xrightarrow[k\to\infty]{} 0$). Finally, observe that $\beta = 2^{\log\beta} = 2^{0.753725\ldots}$.

Theorem 7 (Yao's Minimax Principle)**.** *Let M be an $n \times m$ matrix, the "game", and Δ_n denote the set of all distributions on $[n]$. Then,*

$$\min_{p \in \Delta_n} \max_{q \in \Delta_m} p^{\mathrm{T}} M q = \max_{q \in \Delta_m} \min_{p \in \Delta_n} p^{\mathrm{T}} M q \tag{2}$$

In other terms, if p (resp. q) is the strategy of the first (resp. second) player who aims at minimizing (resp. maximizing) the outcome of a two-stage game with payoffs M, Yao's minimax principle states that if both parties play optimally, the order in which they play does not matter.

We also have the following fact, which essentially says that for any fixed adversarial strategy of the second player, the best and worst expected payoffs for the first player are achieved by deterministic ("pure") strategies:

Fact 8. *Given $q \in \Delta_m$,*

$$\min_{p \in \Delta_n} p^{\mathrm{T}} M q = \min_{i \in [n]} {e_i}^{\mathrm{T}} M q$$
$$\max_{p \in \Delta_n} p^{\mathrm{T}} M q = \max_{i \in [n]} {e_i}^{\mathrm{T}} M q$$

This can be applied to $R(f)$, in order to derive a (simpler) way to get lower bounds on this quantity: for fixed Boolean f on n variables, denote by $\Delta_{\mathcal{T}_f}$ the set of all distributions over $\mathcal{T}_f$, and see the function $\mathrm{cost}\colon (x, T) \in \{0,1\}^n \times \mathcal{T}_f \mapsto \mathrm{cost}(x, T)$ as an $N \times M$-matrix with $N = 2^n$ and $M = |\mathcal{T}_f|$ (recall that $\mathcal{T}_f$ is finite), where the entry indexed by (x, T) contains $\mathrm{cost}(x, T)$. Unraveling the definitions,

$$\begin{aligned}
R(f) &= \min_{p \in \Delta_{\mathcal{T}_f}} \max_{x \in \{0,1\}^n} \mathbb{E}_{T \sim p} \mathrm{cost}(x, T) = \min_{p \in \Delta_{\mathcal{T}_f}} \max_{x \in \{0,1\}^n} \sum_{T \in \mathcal{T}_f} p(T)\, \mathrm{cost}(x, T) \\
&= \min_{p \in \Delta_{\mathcal{T}_f}} \max_{q \text{ over } \{0,1\}^n} \sum_{T \in \mathcal{T}_f} p(T) q(x)\, \mathrm{cost}(x, T) && \text{(Fact 8)} \\
&= \max_{q \text{ over } \{0,1\}^n} \min_{p \in \Delta_{\mathcal{T}_f}} \sum_{T \in \mathcal{T}_f} p(T) q(x)\, \mathrm{cost}(x, T) && \text{(Yao's Minimax Principle)} \\
&= \max_{q \text{ over } \{0,1\}^n} \min_{T \in \mathcal{T}_f} \sum_{T \in \mathcal{T}_f} p(T) q(x)\, \mathrm{cost}(x, T) && \text{(Fact 8)} \\
&= \max_{q \text{ over } \{0,1\}^n} \min_{T \in \mathcal{T}_f} \mathbb{E}_{x \sim q} \mathrm{cost}(x, T) \\
&\geq \min_{T \in \mathcal{T}_f} \mathbb{E}_{x \sim q} \mathrm{cost}(x, T) \qquad \forall q \text{ distribution over } \{0,1\}^n
\end{aligned}$$

In other terms, we reduced the problem back to *deterministic* decision trees; to prove a lower bound, it's only necessary to come up with *one* particularly "bad" distribution q on instances, which fools all deterministic DT.

This approach motivates the definition of the following quantity:

Definition 9 (Distributional DT complexity)**.** *For any distribution q on $\{0,1\}^n$,*

$$D_q(f) \stackrel{\text{def}}{=} \min_{T \in \mathcal{T}_f} \mathbb{E}_{x \sim q} \mathrm{cost}(x, T) \tag{3}$$

Main idea we switched from *worst-case input for randomized DT* to *worst-case distribution on inputs for deterministic DT*:

$$\boxed{R(f) \geq D_q(f) \qquad \forall q} \tag{4}$$

2.3 Application: first lower bound

With this trick up our sleeve, we are ready to prove the following theorem:

Theorem 10. *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a non-trivial monotone function invariant under a transitive permutation group. For simplicity[3], also assume f is balanced: $\mathbb{P}_{x\sim\mathcal{U}_{\{0,1\}^n}}\{\, f(x)=0\,\} = \mathbb{P}_{x\sim\mathcal{U}_{\{0,1\}^n}}\{\, f(x)=1\,\}$. Then,*

$$R(f) = \Omega\Big(n^{2/3}\Big)$$

Observation 11. *Removing the extra assumption on f being balanced, we get that $R(f) = \Omega\Big(N^{2/3}\Big)$ (where $N = \binom{n}{2}$) for monotone graph properties; it is conjectured to be $\Omega(N)$. Moreover, in its general form (that is, under the milder transitive permutation groups invariance assumption), this bound is known to be tight.*

Proof. We will need the following facts of Boolean Fourier analysis:

Definition 12 (Influence and Variance)**.** *Fix f to be a Boolean function on n variables. For $i \in [n]$, the* influence of the i^{th} coordinate *is defined as* $\mathrm{Inf}_i(f) \stackrel{\text{def}}{=} \mathbb{P}_{x\sim\mathcal{U}_{\{0,1\}^n}}\{\, f(x) = f(x^{\oplus i})\,\}$*; the* total influence of f *is* $\mathbb{I}(f) \stackrel{\text{def}}{=} \sum_{i=1}^n \mathrm{Inf}_i(f)$.

The variance of f *is defined as*

$$\mathrm{Var}\, f \stackrel{\text{def}}{=} \mathbb{E}_x(f(x) - \mathbb{E}_x f(x))^2 = 2\mathbb{P}_{x,y}\{\, f(x) \neq f(y)\,\} \underset{\substack{\text{if } f\\ \text{balanced}}}{=} 1$$

where the expectations and probabilities are uniform over $\{0,1\}^n$.

The argument relies on two main results (which will be proven in the next lecture):

Theorem 13 (O'Donnell, Saks, Schramm, Servedio [OSS05])**.** *For $f\colon \{0,1\}^n \to \{0,1\}$ and T a decision tree that computes f,*

$$\mathrm{Var}\, f \leq \sum_{i=1}^n \delta_i(T)\,\mathrm{Inf}_i(f)$$

where $\delta_i(T)$ is the probability that the i^{th} bit of the input is queried by T; that is, $\delta_i(T) = \mathbb{P}_x\{\, T \text{ queries } x_i \text{ on } x\,\}$. In our case, this yields

$$1 \underset{f\,\text{balanced}}{=} \mathrm{Var}\, f \leq \sum_{i=1}^n \delta_i(T) \underbrace{\frac{\mathbb{I}(f)}{n}}_{\substack{\text{from } f\text{'s}\\ \text{invariance}}} = \frac{\mathbb{I}(f)}{n}\sum_{i=1}^n \delta_i(T) = \frac{\mathbb{I}(f)}{n}\mathbb{E}_{x\sim\mathcal{U}}\Big[\substack{\#\text{ queries on } x\\ \text{made by } T}\Big]$$

[3] This is not necessary; one can adapt the proof using p-biased Fourier analysis to get rid of this extra assumption.

$\mathcal{U}$ being the uniform distribution on $\{0,1\}^n$; and by taking the minimum over T on both sides:

$$1 \leq \frac{\mathbb{I}(f)}{n} D_{\mathcal{U}}(f)$$

In particular, with Equation 4,

$$R(f) \geq D_{\mathcal{U}}(f) \geq \frac{n}{\mathbb{I}(f)} \tag{5}$$

At this point, we will also use another result, opportunely relating the total influence to the distributional decision tree complexity:

Theorem 14 (O'Donnell – Servedio (Theorem 3 of [OS07]))**.** *For monotone f, $\mathbb{I}(f) \leq \sqrt{D_{\mathcal{U}}(f)}$*

Combining Equation 5 and Theorem 14, we get $R(f) \geq D_{\mathcal{U}}(f) \geq n^{2/3}$.

□

References

[OS07] Ryan O'Donnell and Rocco A. Servedio. Learning monotone decision trees in polynomial time. *SIAM J. Comput.*, 37(3):827–844, June 2007.

[OSS05] Ryan O'Donnell, Michael Saks, and Oded Schramm. Every decision tree has an influential variable. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '05, pages 31–39, Washington, DC, USA, 2005. IEEE Computer Society.