CS 388R: Randomized Algorithms Fall 2017

Lecture 4 — 09/12, 2017

Prof. Eric Price *Scribe: Daniel Liang and Ridwan Syed*

NOTE: THESE NOTES HAVE NOT BEEN EDITED OR CHECKED FOR CORRECTNESS

1 Rolling Dice

Here's a fun problem. Suppose you roll a fair 6 sided die over and over until you roll a 6. What is the expected number of rolls before you roll a 6? This can be computed in a straightforward way:

$$\begin{aligned}
\mathbb{E}[\#\text{ of rolls}] &= \sum_{i=1}^{\infty} \frac{1}{6}\frac{5}{6}^{i-1} \\
&= \frac{1}{6}(1 + \frac{5}{6} + \frac{5}{6}^2 + \ldots = \frac{1}{1-\frac{5}{6}} = 6) \\
&\quad + \frac{1}{6}(\frac{5}{6} + \frac{5}{6}^2 + \ldots = 6\frac{5}{6}) \\
&\quad + \frac{1}{6}(\frac{5}{6}^2 + \ldots = 6\frac{5}{6}^2) \\
&\quad \vdots \\
&= 6
\end{aligned}$$

So the expected number of rolls is 6. Here's an alternative way to get this answer. Notice that the die has no memory of previous rolls. In particular, after any number of rolls which are not 6, the expected number of additional rolls before seeing a 6 is equal to the expected total number of rolls before seeing a 6. Thus we have

$$\mathbb{E}[\#\text{ of rolls}] = \frac{1}{6} + \frac{5}{6}(1 + \mathbb{E}[\#\text{ of rolls}]) \Rightarrow \mathbb{E}[\#\text{ of rolls}] = 6$$

Here's an even more fun problem. What is the expected number of rolls before seeing a six conditioned on only rolling even numbers? You might think that the answer is 3. However the answer is actually 3/2.

For a moment consider a different experiment. You roll the die over and over until you see a number other than 2 or 4. By an argument similar to the first problem, the expected number of rolls in this experiment is 3/2. Notice that the expected number of rolls conditioned on the final roll being x (for any $x \in \{1, 3, 5, 6\}$) is still 3/2. Conditioned on the final roll being 6 gives us the random variable we care about. Thus the expected number of rolls before seeing a 6 conditioned on only seeing even numbers is 3/2.

2 Von Neumann Minimax Principle

Suppose Alice and Bob are playing a 2 player game. Alice and Bob each have a finite set of (pure) strategies. Alice plays a strategy $i \in [n]$ and Bob plays a strategy $j \in [m]$. Each strategy pair (i, j) has an associated payoff $a_{i,j} \in \mathbb{R}$. That is, if Alice plays i and Bob plays j, the payoff of the game is $a_{i,j}$. We'll think of Alice trying to maximize the payoff and Bob trying to minimize the payoff. We'll allow the players to select their strategies probabilistically. Alice plays according to (mixed) strategy $p \in (\mathbb{R}_+)^n$ with $\|p\|_1 = 1$ such that $Pr[\text{Alice plays } i] = p_i$. Define $q \in (\mathbb{R}_+)^m$ analogously. Define the game's payoff matrix $A \in \mathbb{R}^{n \times m}$ by $A(i, j) = a_{i,j}$. If we assume Alice and Bob sample from their strategies independently, we can compute the expected payoff by

$$\mathbb{E}[\text{Payoff}] = \mathbb{E}_{i \in [n];\ j \in [m]}[a_{i,j}] = \sum_{i \in [n];\ j \in [m]} p_i q_j a_{i,j} = \sum_{i \in [n]} p_i \cdot (\sum_{j \in [m]} a_{i,j} q_j) = p^T A q$$

Suppose Bob publishes his strategy q, and Alice is allowed to choose her strategy p knowing q. We can then write the expected payoff as the following optimization problem:

$$V_p = \max_p \min_q p^T A q$$

Analogously, suppose Bob can select his strategy q knowing Alice's choice of p. Then we can similarly write the expected payoff as the following optimization problem:

$$V_q = \min_q \max_p p^T A q$$

How are V_p and V_q related? It would seem that the player who selects their strategy after their opponent is in a better position to maximize (or minimize) the expected payoff. Somewhat surprisingly this is not the case!

Theorem 1 (Von Neumann Minimax)**.** *Suppose Alice and Bob play a game with payoff matrix A. Let Alice's (mixed) strategy be p and Bob's (mixed) strategy be q. Then*

$$\max_p \min_q p^T A q = \min_p \max_q p^T A q$$

Thus $V_p = V_q$! [1]. In fact, we can say a bit more. Without loss of generality, the inner minimum in V_p can be taken over pure strategies $j \in [m]$. In other words we can assume that the first player plays deterministically! For any p, q we have

$$\begin{aligned} p^T A q &= \sum_{i,j} p_i q_j a_{i,j} \\ &= \sum_{j \in [m]} q_i (\sum_{i \in [n]} a_{i,j} p_i) \\ &= \sum_{i \in [m]} q_j \mathbb{E}_{i \sim p}[a_{i,j}] \\ &\geq \min_{j \in [m]} \mathbb{E}_{i \sim p}[a_{i,j}] \\ &= \min_{j \in [m]} p^T A e_j \end{aligned}$$

[1] The proof of this version of the theorem follows from the Strong Linear Programming Duality Theorem. Though not difficult, it is beyond the scope of this class. For a proof see [AMS99]

By the above for a fixed p we have

$$\min_q p^T A q \geq \min_{i \in j} p^T A e_j$$

Since Bob can always select q to be a point distribution this inequality is tight. Thus we have

$$\max_p \min_q p^T A q = \max_p \min_{j \in [m]} p^T A e_j$$

By a similar argument, we V_q does not change if Alice is restricted to play deterministically. Thus we have

$$\max_p \min_{j \in [m]} p^T A e_j = \min_q \max_{i \in [n]} (e_i)^T A q \tag{1}$$

3 Yao's Principle

Suppose we are interested in the performance of some algorithm randomized algorithm $\mathcal{A}$ for solving a problem P[2]. It is often convenient to view $\mathcal{A}$ as a distribution on deterministic algorithm. In particular, for any choice of random seed s that may be fed in to $\mathcal{A}$, we can think of a deterministic algorithm $\mathcal{A}_s$ which runs $\mathcal{A}$ with s hard-coded in as an 'advice-string'. Similarly, we can think of a distribution on deterministic algorithms as a randomized algorithm $\mathcal{A}$. We want to understand the cost (e.g. runtime, number of queries, etc...) of any such randomized $\mathcal{A}$ on a worst-case input I. As we outline below, the game-theoretic setup of the previous section gives a convenient way to analyze this.

Alice and Bob play the following two player game. Alice plays an instance I of P as a (pure) strategy, while Bob plays a deterministic algorithm $\mathcal{B}$ as a (pure) strategy. Assume as well that any such $\mathcal{B}$ is correct on all inputs. We'll let the cost incurred by $\mathcal{B}$ on instance I, $c(I, \mathcal{B})$ be the payoff function. As before, Alice tries to maximize the payoff, while Bob tries to minimize the payoff. As for mixed strategies, Alice chooses from a finite set of instances according to her choice of distribution $\mathcal{I}$ and Bob chooses from a finite set of algorithms according his choice of distribution $\mathcal{A}$. Applying Theorem 1, we have

$$\max_{\mathcal{I}} \min_{\mathcal{A}} \mathbb{E}_{I \sim \mathcal{I};\ \mathcal{B} \sim \mathcal{A}}[c(I, \mathcal{B})] = \min_{\mathcal{A}} \max_{\mathcal{I}} \mathbb{E}_{I \sim \mathcal{I};\ \mathcal{B} \sim \mathcal{A}}[c(I, \mathcal{B})]$$

By (1) we can re-write the above as

$$\max_{\mathcal{I}} \min_{\mathcal{B}} \mathbb{E}_{I \sim \mathcal{I}}[c(I, \mathcal{B})] = \min_{\mathcal{A}} \max_{I} \mathbb{E}_{\mathcal{B} \sim \mathcal{A}}[c(I, \mathcal{B})]$$

We immediately obtain the following.

Theorem 2 (Yao's Principle)**.** *For any distribution $\mathcal{I}$ on inputs I and distribution $\mathcal{A}$ on deterministic algorithms $\mathcal{B}$ which don't make errors,*

$$\min_{\mathcal{B}} \mathbb{E}_{I \sim \mathcal{I}}[c(I, \mathcal{B})] \leq \max_{I} \mathbb{E}_{\mathcal{B} \sim \mathcal{A}}[c(I, \mathcal{B})]$$

Yao's principle asserts that the expected runtime of a randomized algorithm on the worst case input is at least the expected runtime of the best deterministic algorithm for any distribution on inputs. As we will see Yao's principle is a useful tool for proving lower bounds on the expected run time for Las Vegas algorithms.

[2]We'll think of P being a problem whose input size is fixed. For example P could be evaluating a game tree of a fixed height.

4 Game Tree Lower Bounds

Recall from the last class that we wanted to prove there are instances for which any Las Vegas algorithm must query at least $n^{0.693}$ leaves in expectation to evaluate a NAND tree with n leaves. If we want to apply Yao's principle, we should first come up with a suitable hard distribution. Let's try to make each node be 1 with probability ρ independently. In particular we want a solution to the equation $(1-\rho) = \rho^2$, since a node is 0 if both its children are 1. It is easy to check that we can take $\rho = \frac{\sqrt{5}-1}{2}$. Let $\mathcal{I}$ be the distribution on inputs for which each leaf is set to 1 independently with probability ρ. A reasonable deterministic algorithm would be to depth first search the tree, and discard all the leaves descending from a node whose value has already been determined. In particular, for any node u whose value the algorithm tries to determine, it first recursively evaluates one of it's children. If that child is 0, it discards the remaining leaves descending from that child. It then determines that u's value is 1, and ignores the second child. Otherwise it evaluates the second child. Such an algorithm is called depth first search with pruning. As it turns out this algorithm is optimal for our distribution $\mathcal{I}$ [Santha95]. We record (but do not prove) this as a lemma.

Lemma 3. *Let $\mathcal{I}$ be the distribution over NAND trees of height h for which each leaf is set to 1 independently with probability $\rho = \frac{\sqrt{5}-1}{2}$. Let $T(h)$ be expected number of leaves queried by a depth first search with pruning algorithm on trees $I \sim \mathcal{I}$. Then, $T(h)$ is the optimal expected number of leaves queried by any zero error randomized algorithm evaluating trees $I \sim \mathcal{I}$.*

Now, we can calculate $T(h)$. Clearly $T(1) = (1-\rho)\cdot 1 + \rho \cdot 2 = 1 + \rho$. Similarly, $T(h) = (1-\rho)\cdot T(h-1) + \rho \cdot (2 \cdot T(h-1)) = (1+\rho)T(h-1)$.

$$\begin{aligned} T(h) &= (1+\rho)^h \\ &= (1+\rho)^{\log_2 n} \\ &= 2^{\log_2(1+\rho)\cdot \log_2 n} \\ &= n^{\log_2(1+\rho)} \\ &\geq n^{0.693} \end{aligned}$$

Thus, applying Lemma 3 along with Yao's Principle, we see that there are instances for which any Las Vegas algorithm must query at least $n^{0.693}$ leaves in expectation to evaluate a NAND tree with n leaves.

References

[AMS99] Noga Alon, Yossi Matias, Mario Szegedy. The Space Complexity of Approximating the Frequency Moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

[Santha95] Miklos Santha. On the Monte Carlo decision tree complexity of read-once formulae *Random Structures and Algorithms*, 6:1, 75-87, 1995.