Randomized Algorithms, Summer 2016 Lecture 12 (4 pages)

Yao's Principle and the Secretary Problem

Instructor: Thomas Kesselheim

Let us consider the following *online selection problem*, in which you have to make commitments before you know all you choices. Suppose you want to buy a house. You go see several houses and (a bit simplifying here) after each visit you have to decide immediately and irrevocably if you want to buy this particular house or if you want to keep on looking – then somebody else will buy it. Another motivation would be that you want to find the love of your life. You start dating and (even more simplifying here) after each first date you have to decide whether you you want to marry this person or if you want to keep looking.

We can model this problem as follows. There are n candidates of values $v_1, \ldots v_n \in \mathbb{R}$, $v_i \geq 0$. You see the values of these candidates in order 1, …, n. After having seen the i-th candidate you can choose to select it or to reject it. The goal is to maximize the value of the candidate that you select.

The way such online problems are usually analyzed in algorithmic theory is to perform a worst-case analysis. That is, our goal is to find an algorithm that performs well on all possible instances. A common way to think about this is to consider a hypothetical adversary that defines the numbers. Note that in this setting, we don't know anything about the values that come later on, not even their range.

Definition 12.1. *A (possibly randomized) online algorithm is α-competitive if for every sequence of values $v_1, \ldots, v_n$. we have $\mathbf{E}[v(\mathrm{ALG})] \geq \alpha \max_i v_i$.*

Observation 12.2. *There is no deterministic online algorithm that is better than 0-competitive for any n.*

Proof. Assume there is an α-competitive deterministic algorithm for $\alpha > 0$. Consider its decisions made on the sequence $v_1 = 1$, $v_2 = \frac{2}{\alpha}$, $v_3 = \ldots = v_n = 0$. It has to select the second candidate, otherwise it is not α-competitive. In particular, this means it rejects the first candidate. Now consider the sequence $v_1 = 1$, $v_2 = \ldots = v_n = 0$. Because the first candidate looks the same, as in the other sequence, it has to be rejected. This implies that $\mathrm{ALG} = 0$. $\square$

Note that this bound relies on the fact that the algorithm never picks the first candidate. An easy fix is to flip a coin and depending on this outcome to pick the first candidate or to wait. This generally gives you something better than 0.

Observation 12.3. *There is a $\frac{1}{n}$-competitive randomized online algorithm.*

Proof. The algorithm is trivial: Draw X uniformly from $\{1, \ldots, n\}$ and pick the candidate that comes at position X. With probability $\frac{1}{n}$, it is the one with the highest value. $\square$

We are actually still not happy with this result. This algorithm does not even look at the values. Can't we do better by being a bit smarter? It turns out that on arbitrary sequences this is impossible. Today, we will learn a technique to show impossibility results for randomized algorithms. One consequence is the fact that this stupid algorithm is indeed the best that we can hope for in terms of the competitive ratio.

1 Yao's Principle

Yao's principle is a very simple, yet powerful tool to prove impossibility results regarding worst-case performance randomized algorithms. We state it for algorithms that always do something correct but the profit or cost may vary. Such algorithms are called *Las Vegas* algorithms. We first state it for minimization problems because this is the usual way.

We assume that we have a class of deterministic algorithms $\mathcal{A}$ and a class of instances $\mathcal{X}$. In order to avoid technicalities, assume that both classes are finite. Algorithm $a \in \mathcal{A}$ on instance $x \in \mathcal{X}$ incurs cost $c(a,x) \in \mathbb{R}$. A randomized algorithm is simply a probability distribution over the set of deterministic algorithms $\mathcal{A}$. So, let A be a randomized algorithm (which is now a random variable), then A's worst-case cost is $\max_{x\in\mathcal{X}} \mathbf{E}\left[c(A,x)\right]$.

Theorem 12.4 (Yao's Principle)**.** *Let A be a random variable with values in $\mathcal{A}$ and let X be a random variable with values in $\mathcal{X}$. Then,*

$$\max_{x\in\mathcal{X}} \mathbf{E}\left[c(A,x)\right] \geq \min_{a\in\mathcal{A}} \mathbf{E}\left[c(a,X)\right] \quad .$$

Proof. Let us first write the expectations as sums over all possible outcomes of X and A.

$$\mathbf{E}\left[c(A,x)\right] = \sum_{a\in\mathcal{A}} \mathbf{Pr}\left[A=a\right] c(a,x) \qquad \text{and} \qquad \mathbf{E}\left[c(a,X)\right] = \sum_{x\in\mathcal{X}} \mathbf{Pr}\left[X=x\right] c(a,x)$$

Now we use that the weighted average of a sequence is always upper-bounded by its maximum value. In our case, the weights are $\mathbf{Pr}\left[X=x\right]$. As $\sum_{x\in\mathcal{X}} \mathbf{Pr}\left[X=x\right] = 1$, we have

$$\max_{x\in\mathcal{X}} \mathbf{E}\left[c(A,x)\right] = \max_{x\in\mathcal{X}} \sum_{a\in\mathcal{A}} \mathbf{Pr}\left[A=a\right] c(a,x) \geq \sum_{x\in\mathcal{X}} \mathbf{Pr}\left[X=x\right] \sum_{a\in\mathcal{A}} \mathbf{Pr}\left[A=a\right] c(a,x) \quad .$$

We can now reorder the sums and get

$$\sum_{x\in\mathcal{X}} \mathbf{Pr}\left[X=x\right] \sum_{a\in\mathcal{A}} \mathbf{Pr}\left[A=a\right] c(a,x) = \sum_{a\in\mathcal{A}} \mathbf{Pr}\left[A=a\right] \sum_{x\in\mathcal{X}} \mathbf{Pr}\left[X=x\right] c(a,x) \quad .$$

Finally, we can use that $\sum_{a\in\mathcal{A}} \mathbf{Pr}\left[A=a\right] = 1$ the same way as above to obtain

$$\sum_{a\in\mathcal{A}} \mathbf{Pr}\left[A=a\right] \sum_{x\in\mathcal{X}} \mathbf{Pr}\left[X=x\right] c(a,x) \geq \min_{a\in\mathcal{A}} \sum_{x\in\mathcal{X}} \mathbf{Pr}\left[X=x\right] c(a,x) = \min_{a\in\mathcal{A}} \mathbf{E}\left[c(a,X)\right] \quad . \qquad \square$$

The analogous statement holds maximization problems, where we have a profit $p(a,x)$ that algorithm a achieves on instance x. By setting $c(a,x) = -p(a,x)$, we get the following corollary.

Corollary 12.5. . *Let A be a random variable with values in $\mathcal{A}$ and let X be a random variable with values in $\mathcal{X}$. Then,*

$$\min_{x\in\mathcal{X}} \mathbf{E}\left[p(A,x)\right] \leq \max_{a\in\mathcal{A}} \mathbf{E}\left[p(a,X)\right] \quad .$$

2 Application to the Selection Problem

Let us now apply Yao's principle to the selection problem. We first consider a simplified variant, in which we are only have with the maximum.

Lemma 12.6. *No randomized algorithm guarantees to select the best candidate with probability more than $\frac{1}{n}$ on sequences on $\{0,1,\ldots,n\}$.*

Proof. As the set of instances $\mathcal{X}$ consider all sequences on numbers $\{0,1,\ldots,n\}$. Let $\mathcal{A}$ be the set of all deterministic online algorithms. That is, the point at which some algorithm a stops a sequence x is deterministic. We denote it by $s(a,x)$.[1] Furthermore, it does not depend on the values x_i for $i > s(a,x)$. (These are the values we haven't seen until making our decision.) We define $p(a,x) = 1$ if a selects the maximum number in x, $p(a,x) = 0$ otherwise.

[1]In principle, algorithms are allowed not to make any decision. In these cases, set $s(a,x) = n$.

Observe that a randomized algorithm A selects the maximum number on a fixed sequence x with probability $\mathbf{E}[p(A,x)]$. We want to show that $\min_{x \in \mathcal{X}} \mathbf{E}[p(A,x)] \leq \frac{1}{n}$. To this end, we use Corollary 12.5. It is enough to show that there is a probability distribution over instances X such that $\max_{a \in \mathcal{A}} \mathbf{E}[p(a,X)] = \frac{1}{n}$

Let $x^{(t)}$ be the sequence defined as

$$x^{(t)} = (1, 2, \ldots, t, 0, \ldots, 0) \ .$$

Intuitively, it is difficult to select the maximum element in this sequence because we do not know t in advance and sequences look alike until it is too late, we have no point of reference to make our selection.

Let us now define a probability distribution over these instances as follows. Let T be drawn uniformly at random from $1, \ldots, n$ and set $X = x^{(T)}$. We now claim that this way

$$\max_{a \in \mathcal{A}} \mathbf{E}[p(a,X)] = \frac{1}{n} \ .$$

Let us consider an arbitrary deterministic algorithm a on the input $x^{(n)} = (1, 2, \ldots, n)$. It will select one number in this sequence; that selection's index we denote be $s := s(a, x^{(n)})$. Now, let us consider some other $x^{(t)}$. If $s \leq t$, then the algorithm will make exactly the same decisions because sequences $x^{(t)}$ and $x^{(n)}$ look the same until position t. If $s > t$, then the algorithm selects 0. Therefore, we have $p(a, x^{(t)}) = 1$ if and only if $s = t$. In combination, this yields

$$\mathbf{E}[p(a,X)] = \mathbf{E}\left[p(a, x^{(T)})\right] = \mathbf{Pr}[T = s] = \frac{1}{n} \ .$$

As this holds for any possible algorithm a, we have shown the claim. □

Theorem 12.7. *No randomized algorithm for the selection problem is α-competitive for $\alpha > \frac{1}{n}$.*

Proof. Suppose there is a randomized algorithm A that is α-competitive for $\alpha = \frac{1}{n} + \epsilon$, $\epsilon > 0$. We use this algorithm to derive a contradiction to Lemma 12.6. Let $M \gg 1$ be a huge number. Given a sequence x on $\{0, 1, \ldots, n\}$, in step i feed $v_i = M^{x_i}$ to the algorithm and mirror the selection choice.

Let v^* denote the maximum value given by this sequence, i.e., $v^* = \max_i v_i = M^{\max_i x_i}$. Observe that $v(ALG)$ is v^* if A stops the sequence at its maximum. Otherwise, it is at most v^*/M. So, we have

$$\mathbf{E}[v(ALG)] \leq v^* \mathbf{Pr}[A \text{ selects maximum element}] + \frac{v^*}{M} \ .$$

If our algorithm is α-competitive, then $\mathbf{E}[v(ALG)] \geq \alpha v^*$. Therefore, we have

$$\frac{1}{n} + \epsilon \leq \mathbf{Pr}[A \text{ selects maximum element}] + \frac{1}{M} \ ,$$

and thus

$$\mathbf{Pr}[A \text{ selects maximum element}] \geq \frac{1}{n} + \epsilon - \frac{1}{M} \ ,$$

Setting $M = \frac{2}{\epsilon}$ then leads to a contradiction. □

3 Random Arrival Order

For the negative result it is essential that candidates arrive in order of increasing value. But what if the elements do not come in a worst-case order? We will now consider the case that first an adversary defines values $v_1, \ldots, v_n$ but then a randomly drawn permutation π is applied before we get to see and select the candidates. This problem is known as the *secretary problem.* To simplify the argument, we assume that all values are distinct, i.e., $v_i \neq v_j$ for $i \neq j$.

Theorem 12.8. *For the secretary problem, there is an algorithm that selects the maximum-weight element with probability* $\frac{1}{e} - \frac{1}{n}$.

Proof. Consider the algorithm that observes the first τ elements in the sequence, without selecting any of these. Afterwards, it selects an element if it is the best one so far.

Without loss of generality, let $v_1 < v_2 < \ldots < v_n$. By this definition, the step in which the maximum-weight element arrives is given as $\pi(n)$ and so on.

Observe that the algorithm succeeds if $\pi(n) > \tau$ and no other element is picked before that round.

$$\mathbf{Pr}\left[\text{correct selection}\right] = \sum_{t=\tau+1}^{n} \mathbf{Pr}\left[\pi(n) = t, \text{no element is picked before round } t\right] \enspace .$$

Let $S_t \subseteq [n]$ be the set of elements that arrive before round t. Among these elements, $\max S_t$ is the one with highest value. Observe that no element is picked before round t if and only if $\pi(\max S_t) \leq \tau$. This gives us

$$\mathbf{Pr}\left[\text{correct selection}\right] = \sum_{t=\tau+1}^{n} \mathbf{Pr}\left[\pi(n) = t\right] \mathbf{Pr}\left[\pi(\max S_t) \leq \tau \mid \pi(n) = t\right] \enspace .$$

It is clear that $\mathbf{Pr}\left[\pi(n) = t\right] = \frac{1}{n}$ but what is $\mathbf{Pr}\left[\pi(\max S_t) \leq \tau \mid \pi(n) = t\right]$? By conditioning on $\pi(n) = t$, the set S_t is a uniformly random subset of size $t-1$ drawn from $n-1$ possible elements. Each possible outcome, gives us a minimum. And this minimum is within the first τ rounds with probability $\frac{\tau}{t-1}$. Very formally, we can write this as

$$\begin{aligned}
&\mathbf{Pr}\left[\pi(\max S_t) \leq \tau \mid \pi(n) = t\right] \\
&= \sum_{M \subseteq \{2,\ldots,n\}} \mathbf{Pr}\left[S_t = M, \pi(\max M) \leq \tau \mid \pi(n) = t\right] \\
&= \sum_{M \subseteq \{2,\ldots,n\}} \mathbf{Pr}\left[S_t = M, \mid \pi(n) = t\right] \mathbf{Pr}\left[\pi(\max M) \leq \tau \mid S_t = M, \pi(n) = t\right] \\
&= \sum_{M \subseteq \{2,\ldots,n\}} \mathbf{Pr}\left[S_t = M, \mid \pi(n) = t\right] \frac{\tau}{t-1} \\
&= \frac{\tau}{t-1} \enspace .
\end{aligned}$$

Note that in this argument it is very crucial that if you condition on $M = S_t$ you have only fixed which elements arrive in rounds $1, \ldots, t-1$ but not their mutual order.

Overall, we now get

$$\mathbf{Pr}\left[\text{correct selection}\right] = \sum_{t=\tau+1}^{n} \frac{1}{n} \frac{\tau}{t-1} \geq \frac{\tau}{n} \int_{\tau}^{n} \frac{1}{x} \mathrm{d}x = \frac{\tau}{n} \ln\left(\frac{n}{\tau}\right) \enspace .$$

Now setting $\tau = \lfloor \frac{n}{e} \rfloor$, gives $\frac{\tau}{n} \ln\left(\frac{n}{\tau}\right) \geq \frac{\frac{n}{e} - 1}{n} \ln\left(\frac{n}{\frac{n}{e}}\right) = \frac{1}{e} - \frac{1}{n}$. □