Algorithms and Uncertainty, Summer 2021 — Lecture 10 (5 pages)

Optimal Stopping: Secretary Problem

Thomas Kesselheim — Last Update: May 6, 2021

Recall the optimal stopping problem: There are n candidates of values $v_1, \dots v_n \in \mathbb{R}$, $v_i \geq 0$. You see one candidate after the other. Immediately after having seen a candidate, you have to decide whether to accept or to reject the candidate.

Last time, we covered the case that the candidates arrive in order $1, \dots, n$. The values v_i are drawn independently from (not necessarily identical) probability distributions, which are known in advance. Today, we will consider a different input model. Nothing is known about the candidates a priori; they are arbitrary. However, they are not revealed in order $1, \dots, n$ but in a random order.

In more detail, an adversary defines values $v_1, \dots, v_n$ and then a randomly drawn permutation π is applied before we get to see and select the candidates. That is, in step t, we get to see the candidate of value $v_{\pi(t)}$. 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 $i \neq j$.

Note that the problem has no obvious representation as a Markov decision process because we know nothing about the values beforehand. As we will see later, some parts of the problem can still be represented by a Markov decision process. But before we get there, we will first devise and analyze a simple algorithm.

1 Threshold Algorithm

Due to the fact that the input arrives in random order, it is reasonable to first observe the sequence a bit and then later use these observations to estimate how good the newly arriving candidates are in comparison to the entire sequence.

In particular, let us consider the following *threshold algorithm*: Observe the first τ elements in the sequence, without selecting any of these. Afterwards, select an element if it is the best one so far.

Theorem 10.1. *For any τ, the threshold algorithm selects the maximum-weight element with probability*

$$\sum_{t=\tau+1}^{n} \frac{1}{n} \frac{\tau}{t-1} \quad .$$

Before proving the theorem, let us first understand its implications. Observe that we can approximate the sum by an integral

$$\sum_{t=\tau+1}^{n} \frac{1}{n} \frac{\tau}{t-1} \geq \frac{\tau}{n} \int_{\tau}^{n} \frac{1}{x} \mathrm{d}x = \frac{\tau}{n} \ln\left(\frac{n}{\tau}\right) \quad .$$

Now setting $\tau = \lfloor \frac{n}{\mathrm{e}} \rfloor$, gives $\frac{\tau}{n} \ln\left(\frac{n}{\tau}\right) \geq \frac{\frac{n}{\mathrm{e}}-1}{n} \ln\left(\frac{n}{\frac{n}{\mathrm{e}}}\right) = \frac{1}{\mathrm{e}} - \frac{1}{n}$. We can also (without loss of generality) assume that n is large because we can add candidates of negligible value, so $\frac{1}{n}$ becomes negligible.

Corollary 10.2. *There is an algorithm that selects the maximum-weight element with probability at least $\frac{1}{\mathrm{e}}$.*

Note that this also implies that $\mathbf{E}\left[v(\mathrm{ALG})\right] \geq \max_i v_i \mathbf{Pr}\left[\text{select best}\right] \geq \frac{1}{\mathrm{e}} v(\mathrm{OPT})$. So, we get a guarantee comparing the algorithm's performance to the offline optimum like in competitive analysis. The important difference is that the expectation is not over any internal randomization of the algorithm but rather over the random draw of the permutation.

Proof of Theorem 10.1. 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 $\pi(1)$ and so on.

Observe that the algorithm succeeds if $\pi(1) > \tau$ and no other element is picked before that round.

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

Let $S_t \subseteq [n]$ be the set of elements that arrive before round t. Among these elements, $\min S_t$ is the one with highest value.[1] Observe that no element is picked before round t if and only if $\pi(\min S_t) \leq \tau$. This gives us

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

It is clear that $\mathbf{Pr}\left[\pi(1) = t\right] = \frac{1}{n}$ but what is $\mathbf{Pr}\left[\pi(\min S_t) \leq \tau \mid \pi(1) = t\right]$? By conditioning on $\pi(1) = t$, 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

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

Note the following crucial observation in this argument: Conditioned 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

$$\mathbf{Pr}\left[\text{select best}\right] = \sum_{t=\tau+1}^{n} \frac{1}{n} \frac{\tau}{t-1} \enspace .$$

□

2 The Optimal Algorithm

After this positive result, we would like to understand whether there is something better that we could do. Our goal will be to maximize the probability that we select the best candidate.

[1] The use of the minimum can be slightly confusing here. It is because smaller indices mean higher values.

We will assume that the algorithm only uses pairwise comparisons. That is, the algorithm will only compare values of candidates but not look at the actual numbers. It is not clear that this is without loss of generality because observing the numbers might reveal additional information.

Theorem 10.3. *For any n, any algorithm that only uses pairwise comparisons selects the maximum-weight element with probability at most*

$$\max_{\tau \in \{0,1,\ldots,n\}} \sum_{t=\tau+1}^{n} \frac{1}{n} \frac{\tau}{t-1} \ .$$

Note that it is actually enough to only show that the algorithm with the highest success probability is a threshold algorithm because Theorem 10.1 states the success probabilities of such algorithms exactly.

To prove the theorem, we will interpret any algorithm which only uses pairwise comparisons as a policy in a suitably chosen Markov decision process. The key will be a smart choice of the state space with only very few states.

In order to define the states, we will use *relative ranks* of the candidates. The relative rank $R_t \in \{1, \ldots, t\}$ of the candidate arriving in step t is the number of candidates that arrive in steps $1, \ldots, t$ and are at least as good as the candidate arriving in step t. That is, $R_t = 1$ means that it is the best so far, $R_t = 2$ that it is the second best, and so forth, up to $R_t = t$, which means that all other candidates up to this point were better. It is not difficult to see that such rank vectors are in one-to-one correspondence with permutations. Drawing the permutation uniformly, we assume that all R_t are independent, R_t is drawn uniformly from $\{1, \ldots, t\}$. We select the best candidate if we stop the sequence at t such that $R_t = 1$ and $R_{t'} > 1$ for $t' > t$. The advantage of this notation is that, in step t, the algorithm knows exactly $R_1, \ldots, R_t$ but not $R_{t+1}, \ldots, R_n$.

Example 10.4. *If $n = 4$ and the arriving candidates have value 18, 42, 30, 50, then $R_1 = 1$, $R_2 = 1$, $R_3 = 2$, $R_4 = 1$.*

Let us see a simple calculation in this notation, which we will need later on.

Lemma 10.5. *For all t, we have*

$$\mathbf{Pr}\left[R_{t+1} > 1, R_{t+2} > 1, \ldots, R_n > 1\right] = \frac{t}{n} \ .$$

Proof. We can argue in two ways. On the one hand, the event $R_{t+1} > 1, R_{t+2} > 1, \ldots, R_n > 1$ means that the best candidate arrives by round t. As we know, the probability for this is $\frac{t}{n}$.

On the other hand, we can also consider the permutation being constructed piece-wise. Some candidates have arrived by t. Now, we draw how the candidate in steps t, $t+1$, and so on compare to these existing candidates. We have

$$\begin{aligned}\mathbf{Pr}\left[R_{t+1} > 1, R_{t+2} > 1, \ldots, R_n > 1\right] &= \mathbf{Pr}\left[R_{t+1} > 1\right] \cdot \mathbf{Pr}\left[R_{t+2} > 1\right] \cdot \ldots \cdot \mathbf{Pr}\left[R_n > 1\right] \\ &= \frac{t}{t+1} \cdot \frac{t+1}{t+2} \cdot \ldots \cdot \frac{n-1}{n} = \frac{t}{n} \ . \end{aligned}$$ □

Proof of Theorem 10.3. In the state space of our Markov decision process, we only have to keep track of three pieces of information. Besides the current time step t and the current relative rank R_t, we store whether a selection has been made and whether this selection is still the best candidate. That is, states are triples of the form (t, R_t, C), where $C \in \{\text{NO}, \text{BEST}, \text{LOST}\}$. Here, NO means that no selection has been made, BEST means that the best candidate so far has been selected, and LOST means that after selecting a candidate, there has appeared a better candidate in the sequence. There are two actions ACCEPT and REJECT; when choosing ACCEPT this decision is recorded in the state. There is a reward of 1 after the final step if the best candidate has been selected, otherwise it is 0. We are interested in the reward which we can obtain starting from state $(1, 1, \text{NO})$.

Example 10.6. *Consider again the case that the arriving candidates have values* 18*,* 42*,* 30*,* 50 *and suppose our policy accepts the second candidate. Then we move through states* $(1, 1, \text{NO}), (2, 1, \text{NO}), (3, 2, \text{BEST}), (4, 1, \text{LOST})$ *and receive a final reward of* 0*.*

We would now like to understand the expected reward of the optimal policy starting from any state (t, R_t, C). To keep notation simple, we skip the remaining time horizon because it is clear from the state and only write $V^*(t, R_t, C)$. We start with the following case distinction:

- If $C = \text{LOST}$, then clearly $V^*(t, R_t, C) = 0$.
- If $C = \text{BEST}$, then $V^*(t, R_t, C) = \mathbf{Pr}\left[R_{t+1} > 1, \ldots, R_n > 1\right]$. This holds independent of any actions chosen.
- So, let us now consider $C = \text{NO}$.
  - In the case $R_t = 1$, we can accept, which will make us win if and only if $R_{t+1} > 1, \ldots, R_n > 1$. If we reject, we move to state $(t+1, R_{t+1}, \text{NO})$, where R_{t+1} is random. Now, we can follow the optimal policy from $(t+1, R_{t+1}, \text{NO})$. That is, the expected reward is given by

$$V^*(t, 1, \text{NO}) = \max\{\mathbf{Pr}\left[R_{t+1} > 1, \ldots, R_n > 1\right], \mathbf{E}_{R_{t+1}}\left[V^*(t+1, R_{t+1}, \text{NO})\right]\} \enspace .$$

  - In the case $R_t > 1$, the reward from accepting is certainly 0. So we have to reject because there is no chance of winning and therefore

$$V^*(t, R_t, \text{NO}) = \mathbf{E}_{R_{t+1}}\left[V^*(t+1, R_{t+1}, \text{NO})\right] \enspace .$$

We combine all these cases and write p_t for $\mathbf{E}_{R_t}\left[V^*(t, R_t, \text{NO})\right]$. By Lemma 10.5, we have $\mathbf{Pr}\left[R_{t+1} > 1, \ldots, R_n > 1\right] = \frac{t}{n}$. So, overall, we get

$$\begin{aligned} p_t &= \mathbf{Pr}\left[R_t > 1\right] \cdot p_{t+1} + \mathbf{Pr}\left[R_t = 1\right] \cdot V^*(t, 1, \text{NO}) \\ &= \mathbf{Pr}\left[R_t > 1\right] \cdot p_{t+1} + \mathbf{Pr}\left[R_t = 1\right] \cdot \max\left\{\frac{t}{n}, p_{t+1}\right\} \\ &= \left(1 - \frac{1}{t}\right) p_{t+1} + \frac{1}{t} \max\left\{\frac{t}{n}, p_{t+1}\right\} \enspace . \end{aligned}$$

Observe that $p_t \geq p_{t+1}$ for any t, so $(p_t)_{t \in \{1,\ldots,n\}}$ is non-increasing. In contrast, $\frac{t}{n} < \frac{t+1}{n}$. So, if $p_{t+1} \leq \frac{t}{n}$, then $p_{t+2} < \frac{t+1}{n}$. So, we know that there has to be a $\tau \in \{1, \ldots, n\}$ such that

$$\begin{aligned} p_t &= \begin{cases} \left(1 - \frac{1}{t}\right) p_{t+1} + \frac{1}{t} p_{t+1} & \text{if } t \leq \tau \\ \left(1 - \frac{1}{t}\right) p_{t+1} + \frac{1}{t} \frac{t}{n} & \text{otherwise} \end{cases} \\ &= \begin{cases} p_{t+1} & \text{if } t \leq \tau \\ \left(1 - \frac{1}{t}\right) p_{t+1} + \frac{1}{n} & \text{otherwise} \end{cases} \end{aligned}$$

This already shows us that the optimal policy uses a threshold. But we can also solve the recursion easily by

$$p_1 = p_{\tau+1} = \frac{1}{n} + \frac{\tau}{\tau+1} p_{\tau+2} = \frac{1}{n} + \frac{\tau}{\tau+1}\frac{1}{n} + \frac{\tau}{\tau+1}\frac{\tau+1}{\tau+2} p_{\tau+3} = \ldots = \sum_{t=\tau+1}^{n} \frac{1}{n}\frac{\tau}{t-1} \enspace .$$

Observe that the first equality holds as we do not accept any of the first τ elements. Further, note that $p_1 = V^*(1, 1, \textsc{no})$. So this proves the theorem. □