Secretary Problem and Prophet Inequalities

Thodoris Tsilivis

June 24, 2024

General problem

Today we will be focusing on different variations of the following simple "game". This game consists of a sequence of n steps. At step i:

1. A box containing reward $X_i \geq 0$ is revealed.
2. We **irrevocably** decide to either:
   (a) **commit** to box i with reward X_i, or
   (b) proceed to the **next step** $i+1$.

   If we **commit** the game **ends**.

The **objective** of this game is to come up with an algorithm that picks the box with the **highest reward** i.e. $X_{max} = \max_{i\in[n]} X_i$.

Intuition suggests this to be non-trivial (if not impossible) with a **deterministic** rule, and one would hope that a **randomized** rule may stand a better chance. For a randomized algorithm the aim would be to maximize the expected reward of the selected box. Ideally one would want to prove a high probability of selecting the maximum reward box (but that is not necessary to prove a bound). Lets formalize:

Why its difficult?

To investigate the difficulty of this problem, lets observe the following two instances:

(1) Box 1 has reward $X_1 = 1$ and box 2 has reward $X_2 = M \gg 1$.

(2) Box 1 has reward $X_1 = 1$ and box 2 has reward $X_2 = \frac{1}{M} \ll 1$.

These two instances illustrate why we **cannot hope for a deterministic algorithm that always outputs the highest reward**. The reason for that is that the information for both instances up to box 1 is identical, which means that any deterministic algorithm should make the same decision at step 1 with either instance as input.

Additionally, these two instances illustrate that **we cannot even get any approximation guarantee with a deterministic algorithm**. All deterministic algorithms that decide to commit to box 1 cannot have an approximation ratio lower than M since in **(1)** that is the optimal choice.

On the other hand all deterministic algorithm that decide to not commit to box 1, are only left with committing to box 2 and as a result cannot have approximation ratio better than M due to **(2)**. Since we can set M to be arbitrarily large we have proven that no deterministic algorithm can get an approximation guarantee for this problem.

Naturally we turn our attention to **randomized** algorithms. The most **naive** one to consider is to have an algorithm that uniformly at random selects an index $i \in [n]$ and always commits to box i. We can prove for this algorithm that:

$$\mathbb{E}[\text{Reward}] = \sum_{i=1}^{n} \mathbb{P}[\text{Selecting box } i] \cdot X_i = \frac{1}{n} \sum_{i=1}^{n} X_i \geq \frac{X_{max}}{n}$$

This randomized algorithm is **super simple and yet it is the best** we could hope for. The intuition for that is the following family of instances:

$$X_i = \begin{cases} M^i & \text{if } i \leq k, \\ 0 & \text{if } i > k. \end{cases}$$

In this family of instances, any *good* algorithm must be able to pick exactly the box at index k really high probability. However since k can be any index in $[n]$, this high probability cannot be higher than $\frac{1}{n}$. Additionally, this whole instance is essentially the same as requiring the algorithm to be able to correctly guess k, which is the same thing that the naive randomized algorithm does. This intuition can be formalized using Yao's minimax principle, and thus to prove that no randomized algorithm can achieve a better than a linear approximation ratio guarantee.

Our discussion thus far has pinpointed **two major hardships** in this problem:

- The **ordering** of the boxes is adversarially picked.
- The **range** of non-negative numbers that our algorithm will be dealing with can be arbitrary.

Lets now see how **relaxing** these two hardships can facilitate significant results.

The Secretary problem

The Secretary Problem is the relaxation of our original game that assumes that the boxes will arrive in **uniformly random order**. That means that we will require our algorithms to perform well enough over potentially adversarially selected rewards X_i, but the boxes will arrive to us in order $X_{\sigma(1)}, X_{\sigma(2)}, \ldots, X_{\sigma(n)}$ where $\sigma(\cdot)$ is just one random permutation of $[n]$ (to avoid notation overload we will keep on using X_i instead of $X_{\sigma(i)}$). Even though this may be counter intuitive, this assumption suffices to design algorithms with great guarantees!

A constant approximation algorithm

A really simple and intuitive algorithm to consider is the following:

1. Observe and reject the first $\frac{n}{2}$ boxes.
2. After that, pick the first box that has reward higher than the previous observed.

To prove the approximation guarantee of this algorithm we will loosely estimate the probability of our algorithm selecting the exact highest reward. One sufficient (but not necessary) event E for that is to have the $2nd$ highest reward box in the first $\frac{n}{2}$ boxes and the highest reward box on the second $\frac{n}{2}$ boxes. Lets define events $E_1 = \{\text{highest reward in second } \frac{n}{2} \text{ boxes}\}, E_2 = \{\text{2nd highest reward in first } \frac{n}{2} \text{ boxes}\}$. The probability of E can thus be analyzed as:

$$\mathbb{P}[E] = \mathbb{P}[E_2] \cdot \mathbb{P}[E_1|E_2] = \frac{n}{2n} \cdot \frac{n}{2n-1} \geq \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$$

By law of total expectation we can thus argue that:

$$\mathbb{E}[\text{Reward}] = \mathbb{E}[\text{Reward}|E] \cdot \mathbb{P}[E] + \mathbb{E}[\text{Reward}|\bar{E}] \cdot \mathbb{P}[\bar{E}] \geq \frac{1}{4} X_{max}$$

The optimal approximation algorithm

To achieve the best possible, we only need to slightly modify our previous algorithm. The fraction of boxes to discard in the initial phase was chosen to facilitate ease of calculation. Now, we will parameterize the number of boxes of the initial phase as m, calculate the probability of selecting the highest reward as a function of m and optimize (over all possible values of m) to get the highest success probability. Towards that end we calculate the following probability:

$$\begin{aligned}
\mathbb{P}[\text{Pick highest reward box}] &= \sum_{t=m+1}^{n} \mathbb{P}[\text{Pick highest at } t] \\
&= \sum_{t=m+1}^{n} \mathbb{P}[X_t \text{ is highest}] \cdot \mathbb{P}[\text{max of } t-1 \text{ is in first } m] \\
&= \sum_{t=m+1}^{n} \frac{1}{n} \cdot \frac{m}{t-1} = \frac{m}{n} \sum_{t=m+1}^{n} \frac{1}{t-1} = \frac{m}{n} \sum_{t=m}^{n-1} \frac{1}{t} \\
&= \frac{m}{n} \left(H_{n-1} - H_{m-1}\right) \\
&\approx \frac{m}{n} \ln\left(\frac{n-1}{m-1}\right)
\end{aligned}$$

The first equality is due to the law of total probability and the fact that the algorithm only selects boxes after the m-th box. The second equality is breaking down the probability of picking the highest reward at t into the independent events that **i)** X_t is in fact the highest reward (of the sequence), **ii)** the 2nd highest reward observed up until box t is part of the sample i.e. is observed before m (if it weren't the algorithm would have terminated when it encountered since the set threshold would be lower).

Selecting m that maximizes this probability is analytically non-trivial. Nonetheless, if we assume that n is large we can calculate that the optimal selection as $m^* = \lceil \frac{n}{e} \rceil$. Assigning m^* we get that the probability that we pick the highest reward box is $\mathbb{P}[\text{Pick highest reward box}] = \frac{1}{e}$ which in turn yields:

$$\mathbb{E}[\text{Reward}] \geq \mathbb{P}[\text{Pick highest reward box}] \cdot X_{max} = \frac{1}{e} X_{max}$$

Finally it can be shown that this result is tight, which means that no algorithm can achieve a better than $\frac{1}{e}$ approximation guarantee. The proof of that relies on a theorem that states that all strategies that maximize the probability of picking the highest reward box can be assumed to be *wait & pick* strategies. The full proof can be found in multiple textbooks (for examples chapter 11 in [1]).

Remember that this algorithm is designed to alleviate the lack of information in this setting. This algorithm achieves some form of "learning" by essentially sacrificing some boxes. These boxes are used to calculate a simple but adequate threshold which in turn will be used to evaluate the remaining boxes. As will become even more evident in the following setting, part of the essence of this "game" is coming up with a good threshold.

The Prophet inequality

In the prophet inequality setting, we are again interested in solving the same problem, but we now have some underlying structure about the range of the rewards that we will need to reason about. Namely, we are assuming that rewards are drawn from distributions $D_1, D_2, \ldots, D_n$.

Since rewards will be drawn from distributions, we switch our benchmark from $\max_{i\in[n]} X_i$ to the more appropriate benchmark of $\mathbb{E}[\max_{i\in[n]} X_i] = \mathbb{E}[X_{max}]$. To convince ourselves that this is necessary, consider if we can select distributions for which the difficult families of instances presented in our introductory section occur with positive probability. We can easily do that, and targeting a benchmark of $\max_{i\in[n]} X_i$ suffers the same hardships we already discussed. In that sense, the newly selected benchmark $\mathbb{E}[\max_{i\in[n]} X_i] = \mathbb{E}[X_{max}]$ requires a weaker form of a guarantee.

The reason why this setting is called *prophet* inequality is that we are comparing ourselves to a "prophet" that can observe all rewards beforehand and always chooses the maximum reward of the boxes (which on a randomly drawn instance is exactly $\mathbb{E}[\max_{i\in[n]} X_i]$). Continuing on our surprising trend, we state the following theorem.

Theorem 1. *(Krengel-Sucheston, 1984, [3]) For every sequence $D_1, D_2, \ldots, D_n$ of independent reward distributions, there exists a threshold rule that guarantees expected reward at least $\frac{1}{2}\mathbb{E}[\max_{i\in[n]} X_i]$.*

A threshold rule, is one where the algorithm simply needs to set a threshold T and accept the first box that has a reward higher than T. In a sense, the secretary algorithms we discussed before were in the same spirit, with the initial phases serving the purpose of estimating a suitable T.

Distributional access

Assuming distributions for our rewards and having access to these distributions means that we do not need to waste any part of our input to set a good threshold.

Lets now discuss the expected reward of an algorithm that sets an arbitrary threshold T. To do so we define the probability of committing to any of the boxes as $p(T)$ (i.e. the probability of at least one box having reward higher than the threshold), and the random variable X_{alg} that denotes the reward that the algorithm will receive (which could potentially be 0). Additionally we define the following function:

$$(x)^+ = \begin{cases} x & \text{if } x \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

Thus we can now analyze as:

$$\mathbb{E}[X_{alg}] = \mathbb{E}[X_{alg} - T + T] = \mathbb{E}[X_{alg} - T] + T \cdot p(T)$$

To proceed we further analyze the first term as follows:

$$\begin{aligned}
\mathbb{E}[X_{alg} - T] &= \sum_{i=1}^{n} \mathbb{E}[X_i - T | X_i \text{ is the first reward} > T] \cdot \mathbb{P}[X_i \text{ is the first reward} > T] && (1)\\
&\geq \sum_{i=1}^{n} \mathbb{E}[X_i - T | X_i \geq T, X_j < T \ \forall j \neq i] \cdot \mathbb{P}[X_i \geq T, X_j < T \ \forall j \neq i] && (2)\\
&\geq \sum_{i=1}^{n} \mathbb{E}[X_i - T | X_i \geq T] \cdot \mathbb{P}[X_i \geq T] \cdot \mathbb{P}[X_j < T \ \forall j \neq i] && (3)\\
&\geq \sum_{i=1}^{n} \mathbb{E}[X_i - T | X_i \geq T] \cdot \mathbb{P}[X_i \geq T] \cdot (1 - p(T)) && (4)\\
&\geq (1 - p(T)) \cdot \sum_{i=1}^{n} \mathbb{E}[(X_i - T)^+] && (5)\\
&= (1 - p(T)) \cdot \mathbb{E}[\sum_{i=1}^{n} (X_i - T)^+] && (6)\\
&\geq (1 - p(T)) \cdot (\mathbb{E}[X_{max}] - T) && (7)
\end{aligned}$$

The first equality (1) is due to the law of total expectation. Inequality (2) relaxes event $\{X_i \text{ is the first reward} > T\}$ to a subset event $\{X_i \geq T, X_j < T \ \forall j \neq i\}$. Inequality (3) further relaxes events and inequality (4) uses the definition of $p(T)$. Finally, inequality (5) uses the definition of $(\cdot)^+$, equality (6) uses linearity of expectations and equality (7) uses the fact that the sum of multiple non-negative terms is always greater than the highest of these terms and the fact that $(x)^+ \geq x$.

Putting everything together we have:

$$\mathbb{E}[X_{alg}] \geq (1 - p(T)) \cdot (\mathbb{E}[X_{max}] - T) + T \cdot p(T)$$

There are two ways to select T such that we get the 2-approximation guarantee.

One way to set the threshold T (due to Samuel-Cahn, 1984 [5]) is by setting a threshold equal to the median of $\max_i X_i$ (i.e. a value T such that $p(T) = \frac{1}{2}$). This yields:

$$\begin{aligned}
\mathbb{E}[X_{alg}] &\geq \frac{1}{2} \cdot (\mathbb{E}[X_{max}] - T) + T \cdot \frac{1}{2}\\
&= \frac{1}{2} \cdot \mathbb{E}[X_{max}]
\end{aligned}$$

Another way to set the threshold T (due to Kleinberg and Weinberg, 2012, [2]) is by directly setting $T = \frac{1}{2} \cdot \mathbb{E}[X_{max}]$. This yields:

$$\begin{aligned}\mathbb{E}[X_{alg}] &\geq (1-p(T)) \cdot (\mathbb{E}[X_{max}] - \frac{1}{2}\mathbb{E}[X_{max}]) + \frac{1}{2}\mathbb{E}[X_{max}] \cdot p(T) \\ &= \frac{1}{2}\mathbb{E}[X_{max}] - \frac{1}{2} \cdot p(T) \cdot \mathbb{E}[X_{max}] + \frac{1}{2} \cdot p(T) \cdot \mathbb{E}[X_{max}] \\ &= \frac{1}{2} \cdot \mathbb{E}[X_{max}]\end{aligned}$$

To further emphasize this result, it is important to state that it is in fact tight i.e. we cannot get and algorithm that achieves a better approximation ratio guarantee. One family of instances that showcases this are instances with two boxes, and distributions with deterministic $X_1 = 1$ and $X_2 = \frac{1}{\epsilon}$ with probability ϵ and 0 otherwise. The expected maximum can be calculates to be $\mathbb{E}[X_{max}] = 2 - \epsilon$ whilst any randomized algorithm that picks box 1 with probability $p_1 \in [0, 1]$ and box 2 with the remainder probability gets $\mathbb{E}[X_{alg}] = p_1 + (1 - p_1) \cdot \epsilon \cdot \frac{1}{\epsilon} = 1$.

Sample Access

We are now switching up the access that the algorithm has to the distributions to samples. Intuitively one would expect that this makes the problem much much harder, since the information available to the algorithm is reduced significantly and is subject to randomization with potentially high variance. Surprisingly, we will now present an threshold rule that only uses one sample per distribution and achieves the same approximation guarantee as the rules in the previous section. The analysis and proofs for sample access to the distributions are all due to [4]. Lets formally state their most interesting result:

Theorem 2. *The algorithm that takes as input samples $\tilde{X_1}, \tilde{X_2}, \dots \tilde{X_n}$ drawn independently from $D_1, D_2, \dots, D_n$ and sets threshold $T = \max_{i \in [n]} \tilde{X_i}$ yields expected reward $\frac{1}{2}\mathbb{E}[X_{max}]$.*

Before we proceed with the analysis for this result, it is important to take a step back to realize what this entails. We already discussed that the threshold algorithm for the prophet inequality setting is tight, but notice that this approach has another significant benefit, computational simplicity (and tractability). Computing either $\mathbb{E}[\max_{i \in [n]} X_i]$ or $T : p(T) = \frac{1}{2}$ can range from non-trivial up to analytically infeasible even for simple distributions. On the other hand, the threshold rule that uses the samples can be computed in a simple linear pass.

Lets now proceed to analyze the expected reward of the algorithm from Theorem 2. To begin the analysis, it is required to formalize the process of sample generation using the **principle of deferred decisions**. That can be described as:

1. Draw two samples from each of $D_1, \dots, D_n$ independently.
2. Denote the larger draw of D_i as Y_i and the small draw of D_i as Z_i (in other words $Y_i > Z_i$).
3. Flip a fair coin independently for each D_i . If coin i is heads, set $X_i = Y_i$ and $\tilde{X_i} = Z_i$. Otherwise, set $X_i = Z_i$ and $\tilde{X_i} = Y_i$.

Notice now that this procedure correctly generates samples $\tilde{X_1}, \dots \tilde{X_n}$ and rewards $X_1, \dots, X_n$ as independent draws from $D_1, \dots, D_n$.

Consider drawn values $Y_1, \ldots, Y_n$ and $Z_1, \ldots, Z_n$ as described above. For the analysis we will require to take all these values (Y_i's and Z_i's) in one single list, sort them in descending order and relabel them as $W_1, \ldots, W_{2n}$. For bookkeeping purposes we need to also track from what distribution W_j came from. To this end we will say that W_j *comes from* i if $W_j = Y_i$ or $W_j = Z_i$. It is helpful to point out that variable W_1 will take some Y value (say Y_k) (and similarly W_{2n} will take a Z value). Finally we will define index j^* to be the index in the sequence of W values that corresponds to the highest Z value, i.e. $W_{j^*} = \max_{i \in [n]} Z_i$. This is a natural barrier for the analysis of both the expected maximum and the expected reward of the algorithm, since at least one W_j value with $j \in [1, j^*]$ is guaranteed to be in the samples $\tilde{X}_1, \ldots, \tilde{X}_n$ and at least one other $W_{j'}$ with $j' \in [1, j^*]$ is guaranteed to be in the drawn rewards $X_1, \ldots X_n$.

Notice that preliminaries so far correspond to the draws from the distributions and we need to also consider the coin drops in step 3 to analyze both the benchmark and the expected reward of the algorithm. To that end, we will define C_j to correspond to the coin flip associated with W_j (i.e. the coin flip that decides if W_j (which can be either the Y_i value or the Z_i value from some distribution D_i) will be in the samples $\tilde{X}_1, \ldots, \tilde{X}_n$ or in the boxes $X_1, \ldots X_n$). Crucially notice that coin flips are independent for each distribution which means that if W_j comes from i and $W_{j'}$ comes from j' with $j \neq j'$ then C_j and $C_{j'}$ are independent. We can now argue that all draws in $C_1, C_2, \ldots, C_{j^*-1}$ are mutually independent **but** C_{j^*} is not independent of these draws. That is because if W_{j^*} comes from distribution i then there exists one index $j' \in [1, j^* - 1]$ such that $W_{j'}$ also comes from distribution i (since we sorted in descending order). We are now ready to proceed with the proof.

Firstly, we will calculate the benchmark i.e. the expected maximum in this setting (the prophet's payout). To that end remember that if coin toss i lands on heads then we will get the Y_i value in a reward box (which is always favorable for the prophet who selects the offline maximum reward). We can now argue that:

$$X_{max} = \max_{i \in n} X_i = \begin{cases} W_j & \text{w.p. } \frac{1}{2^j} \text{ for } j < j^*, \\ W_{j^*} & \text{w.p. } \frac{1}{2^{j^*-1}} \end{cases}$$

The argument for this is that we can decompose into independent events in the following way (notice that the decomposition covers the entire state space, i.e. the probabilities add up to 1):

- $X_{max} = W_1$ if C_1 is heads (probability $\frac{1}{2}$),
- $X_{max} = W_2$ if C_1 is tails and C_2 is heads (probability $\frac{1}{2^2}$),
- ...
- $X_{max} = W_{j^*-1}$ if $C_1, C_2, \ldots C_{j^*-2}$ are tails and C_{j^*-1} is heads (probability $\frac{1}{2^{j^*-1}}$),
- $X_{max} = W_{j^*}$ if $C_1, C_2, \ldots C_{j^*-1}$ are tails (remember that C_{j^*}) (probability $\frac{1}{2^{j^*-1}}$),

Thus we can calculate the expected maximum exactly as:

$$\mathbb{E}[X_{max}] = \sum_{j=1}^{j^*-1} \frac{W_j}{2^j} + \frac{W_{j^*}}{2^{j^*-1}}$$

Now we need to reason about the the expected reward that the algorithm will get. The analysis will similarly break down the space into favorable disjoint events:

- If C_1 is tails then the algorithm gets 0 reward.
- If for $j < j^* - 1$ we have that $C_1, C_2, \ldots, C_j$ are heads and C_{j+1} is tails then we can argue that the algorithm will pick a box that has reward at least W_j (since W_{j+1} is threshold, and there exist boxes that have reward larger than that). The probability of this event is $\frac{1}{2^{j+1}}$
- If $C_1, C_2, \ldots, C_{j^*-1}$ are all heads then the algorithm will get reward at least W_{j^*-1}. The probability of this event is $\frac{1}{2^{j^*-1}}$

We can now bound the expected reward of the algorithm and prove the theorem:

$$\begin{aligned}
\mathbb{E}[Reward] &\geq \sum_{j=1}^{j^*-2} \frac{W_j}{2^{j+1}} + \frac{W_{j^*-1}}{2^{j^*-1}} \\
&= \sum_{j=1}^{j^*-2} \frac{W_j}{2^{j+1}} + \frac{1}{4} \cdot \frac{W_{j^*-1}}{2^{j^*-1}} + \frac{3}{4} \cdot \frac{W_{j^*-1}}{2^{j^*-1}} \\
&\geq \sum_{j=1}^{j^*-1} \frac{W_j}{2^{j+1}} + \frac{W_{j^*}}{2^{j^*}} \\
&= \frac{1}{2} \left(\sum_{j=1}^{j^*-1} \frac{W_j}{2^j} + \frac{W_{j^*}}{2^{j^*-1}} \right) \\
&= \frac{1}{2} \mathbb{E}[X_{max}]
\end{aligned}$$

where we used simple algebra and the formula of the expected maximum that we computer beforehand.

Conclusions

In these notes we introduced and analyzed an optimal stopping problem. Since this problem is provably impossible to deal with, we drew assumptions that allowed for meaningful guarantees. The secretary problem arose when assuming that the order of the arrivals is uniformly random, whilst the prophet inequality setting arose when we assumed that rewards were drawn from distributions. The problem has significant real world applications, most notably auctions, and has thus been thoroughly studied in the past couple of decades, including numerous combinatorial extensions of it.

References

[1] *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, 2021.

[2] Robert Kleinberg and S. Matthew Weinberg. "Matroid Prophet Inequalities". In: *CoRR* abs/1201.4764 (2012). arXiv: 1201.4764. URL: http://arxiv.org/abs/1201.4764.

[3] Ulrich Krengel and Louis Sucheston. "Semiamarts and finite values". In: *Bulletin of The American Mathematical Society - BULL AMER MATH SOC* 83 (Oct. 1977). DOI: 10.1090/S0002-9904-1977-14378-4.

[4] Aviad Rubinstein, Jack Z Wang, and S Matthew Weinberg. "Optimal single-choice prophet inequalities from samples". In: *arXiv preprint arXiv:1911.07945* (2019).

[5] Ester Samuel-Cahn. "Comparison of threshold stop rules and maximum for independent nonnegative random variables". In: *the Annals of Probability* (1984), pp. 1213–1216.