CS787: Advanced Algorithms	**Scribe:** Maulik Shah
Lecture 16: Prophet Inequlities, Online Predictions	**Date:** 3/28/19

16.1 Previous Lecture

The topic covered in the previous lecture was the problem of interviewing n candidates in a random order, with the goal of maximizing the probability of hiring the best one. such that it can be achieved with a probability $1/e$ but nothing better than that.

16.2 This Lecture

In this lecture we discuss the case where each candidate has a quality which is represented by a random variable X_i with known distribution. One way to think of this is to imagine having the resumes of every candidate before the interviews, so we have an idea of what to expect (distribution of X_i), but when we conduct the interview, that idea gets grounded in reality (getting a sample x_i of X_i).

Our goal in this one is to maximize the quality of the hire subject to randomness in the input as well as the algorithm. Note that once a candidate is rejected, you cannot go back and hire them. So you have to make a decision of whether to hire them or not at the moment you first see them, this is what makes it an online algorithm.

For example, consider a case where you interview candidate 2. This means you are getting a sample x_2 of X_2. After looking at this value, you need to decide whether to hire them or not. If yes, then the algorithm stops. If no, the algorithm continues with sampling x_3 from the distribution of X_3 and so on.

The optimal hindsight strategy would be hiring the candidate which has the highest quality. This hindsight strategy hires the candidate with expected quality $E[\max_{i\in[n]} X_i]$. This value is referred to as the hindsight optimum.

For example, consider two candidates, 1 and 2, with the following probability distributions for their "qualities":

$$X_1 = 1 \text{ with probability } 1,$$
$$X_2 = \begin{cases} 0 & \text{with probability } 9/10 \\ 10 & \text{with probability } 1/10 \end{cases}.$$

Now, the hindsight optimum in this case would be $9/10\times1+1/10\times10 = 1.9$; but for the algorithm, if it decides to hire candidate 1, it will surely get quality 1, and if it hires candidate 2, the expected quality it can get is still $10\times1/10 = 1$. So in this case, irrespective of what the algorithm does, the result is the same.

Now, suppose the order of arrival of candidates is fixed and known, then we can use a dynamic

programming algorithm (DP) to solve this in the following manner:

Suppose DP computes V_i = expected quality obtained by the optimal online policy for hiring one of the candidates $i, i+1, \ldots, n$.

Let $V_i(x_i)$ be the better of the two choices among hiring candidate i or not hiring candidate i, i.e., $V_i(x_i) = \max(x_i, V_{i+1})$. Then $V_i = \mathrm{E}_{x_i \sim X_i}[\max(x_i, V_{i+1})]$.

Now, we can compute the expected quality of the candidate hired by this algorithm by backward induction, starting from candidate n.

If you only have one candidate n, you have no other option than to hire candidate n, as you won't get anything if you reject them.

If you have two candidates $n-1$ and n, the decision would be based on considering the maximum from the current instantiation x_{n-1} of X_{n-1}, and V_n, which is nothing but the expected value of X_n.

Following a similar argument, the recursive relationship $V_i = \mathrm{E}_{x_i \sim X_i}[\max(x_i, V_{i+1})]$ holds.

Consider now the following example:

$$
\begin{aligned}
&X_1 = 1 \text{ with probability } 1, \\
&X_2 = \begin{cases} 0 & \text{with probability } 99/100 \\ 100 & \text{with probability } 1/100 \end{cases}, \\
&X_3 = \begin{cases} 0 & \text{with probability } 999/1000 \\ 1000 & \text{with probability } 1/1000 \end{cases}.
\end{aligned}
$$

Now, the expected return on selecting the candidates, based on DP, are as follows:

$$
\begin{aligned}
&V_3 = E[X_3] = 1, \\
&V_2 = E_{x_2 \sim X_2}[\max(x_2), V_3] = 100 \times 1/100 + 1 \times 99/100 = 1.99, \\
&V_1 = E_{x_1 \sim X_1}[\max(x_1), V_2] = 1.99.
\end{aligned}
$$

However, this method only works when the order of arrival of the candidates is known in advance. Also, we have no idea how the performance of this algorithm compares against the hindsight optimum. Next, we consider a general class of policies called threshold policies.

16.3 Threshold Policies

These policies determine a threshold τ based on the distributions of $X_1, X_2, \ldots, X_n$ and hires the first ever candidate with quality at least τ. (It is possible that you end up hiring no one.)

Let us consider the following threshold. Define τ to be the value t such that

$$\Pr[\exists\ i \text{ with } X_i \geq t] = \frac{1}{2}.$$

In other words, $\tau = \text{median}(\max_i X_i)$. (There can be multiple values of τ satisfy this definition.)

We now show that this gives us a competitive ratio of 2. On the one hand,

$$\mathrm{OPT} = \mathrm{E}[\max_i X_i] = \tau + \mathrm{E}[\max_i (X_i - \tau)] \leq \tau + \mathrm{E}[\max_i (X_i - \tau)^+].$$

On the other hand, for the value ALG returned by the algorithm we have

$$\begin{aligned}
\mathrm{ALG} &\geq \tau \mathrm{Pr}[\max_i X_i \geq \tau] + \sum_i \mathrm{E}[(X_i - \tau)^+] \mathrm{Pr}[\forall j < i, X_j < \tau] \\
&\geq \tau/2 + \sum_i \mathrm{E}[(X_i - \tau)^+] \mathrm{Pr}[\max_i X_i < \tau] \\
&= \tau/2 + \frac{1}{2} \sum_i \mathrm{E}[(X_i - \tau)^+].
\end{aligned}$$

Consider these two bounds together, we get $\mathrm{ALG} \geq \frac{1}{2}\mathrm{OPT}$, i.e., the competitive ratio is at most 2.

Similar results hold for setting the threshold τ such that

$$\tau = \sum_i \mathrm{E}[(X_i - \tau)^+]$$

or setting

$$\tau = \frac{1}{2}\mathrm{E}[\max_i X_i].$$

This problem was studied first in the context of optimal stopping policies, and they used to refer to the hindsight optimum value as the prophet value.

16.4 Online Prediction

Consider now a different problem of predicting the weather. Given n experts in indexed by $i \in [n]$. On each day t for $t \in [T]$, each expert i makes prediction of rain or shine. Let m_i denote the total number of mistakes made by expert i. And our goal is to design an algorithm that also predicts the weather for each day such that the "regret":= $M - \min_i m_i$ is as small as possible where M is the number of mistakes made by our algorithm. (Ideally, we would like to have an $o(T)$ regret.)

In the next lecture, we will study algorithms for general online prediction problems.