Georgia Tech S'22 CS 6550/8803: Advanced Algorithms & Uncertainty

Lecture 13: Prophet Inequality and Online Contention Resolution

Lecturer: *Sahil Singla* Last updated: March 4, 2022

In the next few lectures we will work with Stochastic Uncertainty, i.e., where the input to your algorithm is drawn from some probability distributions. We will assume that the distributions are known to the algorithm but the exact outcomes/realizations of the random variables are only gradually revealed. The current lecture is about Prophet Inequalities, which are online problems with stochastic inputs. En-route we will learn about the technique of Online Contention Resolution Scheme.

1 Prophet Inequality

We start with the classical single-item Prophet Inequality (PI) problem, which is a stochastic variant of the Online Max Selection from Lecture 8. In this problem we are given[1] distributions $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n$ of n independent non-negative random variables (think of values of n arriving bidders). At the t-th step, the algorithm is revealed random value $X_t \sim \mathcal{D}_t$ and it has to immediately and irrevocably accept/reject this value. Overall, the algorithm can accept at most one of these values (think of selling a single item), say it accepts X_τ, and the goal of the algorithm is to maximize $\mathbb{E}[X_\tau]$.

Inspired by our Competitive Ratio framework that we saw earlier for online problems, we want to maximize the value of our algorithm as compared to the offline optimum. (In the next lecture we will see another benchmark.) In PI, the offline optimum is the random variable $\max_t X_t$, so we define our benchmark to be $\mathbb{E}[\max_t X_t]$. Now the competitive ratio of an online algorithm is the ratio

$$\frac{\mathbb{E}[\max_t X_t]}{\mathbb{E}[X_\tau]}.$$

What is the smallest possible competitive-ratio achievable by an online algorithm?

1.1 Hardness

We first prove a lower bound on the competitive ratio of any online algorithm.

Lemma 1. *No online algorithm can achieve a competitive-ratio which is a constant smaller than* 2.

Proof. Consider $n = 2$ and following distributions: $\mathcal{D}_1$ is a point-mass distribution where $X_1 \sim \mathcal{D}_1$ equals 1 with probability 1; distribution $\mathcal{D}_2$ is a weighted-Bernoulli distribution

[1] We will ignore computational (running-time) aspects today, so it doesn't matter how the distributions are given. However, most of the results today can be made poly-time for explicitly given distributions.

where $X_2 \sim \mathcal{D}_2$ takes a large value $1/\epsilon$ w.p. ϵ and is 0 otherwise, where $\epsilon \to 0$. We first calculate the expected offline optimum:

$$\mathbb{E}[\max_t X_t] \;=\; \epsilon \cdot \frac{1}{\epsilon} + (1-\epsilon)\cdot 1 \;=\; 2-\epsilon.$$

Next, we observe that every online algorithm has expected value $\mathbb{E}[X_\tau] \le 1$. This is because any online algorithm has to decide whether to accept/reject $X_1 = 1$ without looking at the outcome X_2. Thus, the algorithm either accepts $X_1 = 1$ and has $X_\tau = 1$, or if the algorithm rejects X_1 then it can only accept X_2 where we have $\mathbb{E}[X_2] = 1$ (since X_2 is non-zero only with probability ϵ). This proves that any online algorithm has $\mathbb{E}[X_\tau] \le 1$, which means the competitive ratio is at most $(2-\epsilon)/1$, which approaches 2 as $\epsilon \to 0$. □

1.2 Fixed-Threshold Algorithm

We will now design an online algorithm that achieves a competitive ratio of 2 for the single-item PI. The first such result is due to Krengel, Sucheston, and Garling in the late 1970s. We will instead consider the simple *Mean-Threshold* algorithm: compute the threshold-price $P := \frac{1}{2}\mathbb{E}[\max_t X_t]$ using the given distributions and then select the first $X_t \ge P$, i.e., accept the first bidder who is willing to pay the price P.

Theorem 2 ([3])**.** *The Mean-Threshold algorithm has a competitive ratio of* 2*.*

Proof. For any random variable X, let X^+ denote $\max\{X, 0\}$. We start by observing a simple inequality:

$$\begin{aligned}\mathbb{E}[\max_t X_t] \le \mathbb{E}[P + (\max_t X_t - P)^+] &= P + \mathbb{E}[\max_t (X_t - P)^+] \\ &\le P + \mathbb{E}[\sum_t (X_t - P)^+] = P + \sum_t \mathbb{E}[(X_t - P)^+]. \quad (1)\end{aligned}$$

Next, we simplify the expression for the expected value of the Mean-Threshold algorithm. Let $q := \Pr[\max_t X_t \ge P] = 1 - \Pr[\forall_t X_t < P]$ denote the probability that the algorithm accepts some value. Since the algorithm only accepts values above P, we can decompose its value X_τ into value below P and value above P, to get

$$\begin{aligned}\mathbb{E}[X_\tau] &= \sum_t \Pr[\forall_{s<t} X_s < P] \cdot \Pr\left[X_t > P \mid \forall_{s<t} X_s < P\right] \cdot \mathbb{E}[X_t \mid X_t > P, \forall_{s<t} X_s < P] \\ &= \sum_t \Pr[\forall_{s<t} X_s < P] \cdot \Pr\left[X_t > P\right] \cdot \mathbb{E}[P + (X_t - P)^+ \mid X_t > P]\end{aligned}$$

since X_t is independent of X_s for $s < t$. Noticing $q = \sum_t \Pr[\forall_{s<t} X_s < P] \cdot \Pr\left[X_t > P\right]$,

$$\begin{aligned}\mathbb{E}[X_\tau] &= q \cdot P + \sum_t \Pr[\forall_{s<t} X_s < P] \cdot \mathbb{E}\left[(X_t - P)^+\right] \\ &\ge q \cdot P + \sum_t (1-q) \cdot \mathbb{E}[X_t - P]^+ \qquad \text{(by defn of } q\text{)} \\ &\ge q \cdot P + (1-q) \cdot \Big(\mathbb{E}[\max_t X_t] - P\Big) \qquad \text{(using (1))} \\ &= P,\end{aligned}$$

which completes the proof. □

1.3 Prophet Inequality Model for Online Packing Problems

The Prophet Inequality problem gives us a powerful way to bypass the $\Omega(n)$ hardness for the Online Max Selection problem from Lecture 8 on Yao's Minimax by making stochastic assumptions. Thus, we can view PI as another way of going "beyond the worst-case" for online problems, akin to the Random-Order model from Lecture 9. We now generalize the PI problem to the *Prophet Inequality* model for online packing problems, which has seen a lot of research in the last decade due to its beyond-the-worst-case nature and due to its applications in mechanism design (see [2] for further details).

Definition 1 (Prophet Inequality model for Packing)**.** *Suppose each element $i \in V$ takes a value $X_i \in \mathbb{R}_{\geq 0}$ independently from some known distribution $\mathcal{D}_i$. These values are presented one-by-one to an online algorithm in an adversarial order. Given a packing feasibility constraint $\mathcal{F} \subseteq 2^V$, the problem is to immediately and irrevocably decide whether to select the next element i, while always maintaining a feasible solution and maximizing the sum of the selected values.*

A c-competitive prophet inequality for $c \geq 1$ means there exists an online algorithm with expected value at least $1/c$ times the expected value of an offline algorithm that knows all values from the beginning. Several packing families admit good PIs: we can be 2-competitive for Matroids [3], $O(1)$-competitive for matchings [5], $O(k)$-competitive for intersection of k matroids [3], and $O(\log^2 n)$-competitive for arbitrary packing constraint $\mathcal{F} \subseteq 2^V$ [4].

2 Prophet Inequalities via an LP Relaxation

Although we already saw a proof of Theorem 2, the proof seems ad-hoc and it's not clear how one would generalize or come-up with it. We will now see a more streamlined way of designing Prophet Inequalities, which goes back to how we would design an approximation algorithm for an NP-hard problem (offline): first write a LP/convex relaxation of the problem and then round the fractional solution.

2.1 LP Relaxation

For simplicity, in the rest of section we will assume that all distributions $\mathcal{D}_t$ are weighted-Bernoulli, i.e., $X_t \sim \mathcal{D}_t$ takes value some known $v_t \geq 0$ w.p. p_t and is 0 otherwise. (In HW-3 we will see how to extend the results to more general distributions.) Note that the hardness of $\geq$ 2-competitive in Lemma 1 continues to hold even for weighted-Bernoulli random variables, so we cannot hope to do anything better than 2-competitive.

We first write an LP that gives us an upper bound on our benchmark of the expected offline

optimum.

$$\begin{aligned} \max_{x} \quad & \sum_t x_t v_t & \text{(LP)} \\ \text{s.t.} \quad & x_t \le p_t \qquad \forall t \\ & \sum_t x_t \le 1 \\ & x_t \ge 0 \qquad \forall t \end{aligned}$$

Let $\mathbf{x}^*$ denote the optimal solution of (LP).

Lemma 3. $\sum_t x_t^* v_t \ge \mathbb{E}[\max_t X_t]$.

Proof. It suffices to give a feasible solution for (LP) with objective value at least $\mathbb{E}[\max_t X_t]$. Define q_t to be the probability that X_t is non-zero and it achieves the maximum in $\max_t X_t$ (assume all v_t-s are distinct so that there are no tie-breaks; otherwise you can add a slight noise to make them distinct). We claim that q_t is a feasible solution to (LP). First, see that $0 \le q_t \le p_t$ since X_t is non-zero with probability at most p_t. Next, we have $\sum_t q_t \le 1$ since for any outcome there is at most one of the X_t-s that achieves $\max_t X_t$. Finally, observe that $\sum_t q_t v_t = \mathbb{E}[\max_t X_t]$ because whenever X_t is non-zero and achieves $\max_t X_t$, it contributes exactly v_t to the maximum. □

Remark: There are examples where the inequality in Lemma 3 can be strict. E.g., suppose n=2 and X_1, X_2 are distributed i.i.d. Bernoulli random variables, i.e., $X_i = 1$ w.p. $1/2$ and 0 otherwise. In this example $\mathbb{E}[\max\{X_1, X_2\}] = 3/4$ but $x_1^* v_1 + x_2^* v_2 = 1$ since $x_1^* = x_2^* = 1/2$. We leave as an exercise to design an example where $\sum_t x_t^* v_t = \frac{e}{e-1}\mathbb{E}[\max_t X_t]$ for $n \to \infty$.

Given Lemma 3, to design a 2-competitive algorithm it suffices to design an online algorithm with $\mathbb{E}[X_\tau] \ge \frac{1}{2}\sum_t x_t^* v_t$. We will design such an algorithm by rounding the fractional variables $\mathbf{x}^*$ in an online way for our PI problem.

2.2 Online Contention Resolution Scheme

At a high level, the optimal solution $\mathbf{x}^*$ to (LP) indicates how frequently we should be accepting the random variables: we want to accept the t-th random variable x_t^* fraction of the times when it takes value v_t. If there was no constraint on how many random variables we can accept, this would be easy: take X_t whenever it takes value v_t independently with probability $\frac{x_t^*}{p_t}$, so that on average we are getting each X_t with probability $\Pr[X_t = v_t] \cdot \frac{x_t^*}{p_t} = x_t^*$ while it takes value v_t.

In the presence of the constraint that we can only accept at most 1 random variable, a natural algorithm would be to accept the next X_t whenever it takes value v_t independently with probability $\frac{x_t^*}{p_t}$, unless we have already accepted something. However, such an algorithm might have a very poor performance: Consider again the example from the proof of Lemma 1: Consider a PI instance for n=2 where $\mathcal{D}_1$ is a point-mass distribution where $X_1 \sim \mathcal{D}_1$ equals 1 with probability 1; distribution $\mathcal{D}_2$ is a weighted-Bernoulli distribution where $X_2 \sim \mathcal{D}_2$ takes a large value $1/\epsilon^2$ w.p. ϵ and is 0 otherwise, where $\epsilon \to 0$. (Note that

the expected offline optimum is $\approx 1/\epsilon$ here.) The optimal solution $\mathbf{x}^*$ for this instance has $\mathbf{x}^* = (1-\epsilon, \epsilon)$. However, an algorithm that always accepts X_1 w.p. $\frac{x_1^*}{p_1} = 1-\epsilon$, reaches X_2 without accepting anything at most ϵ fraction of the times. Hence, this algorithm accepts X_2 while it takes value $1/\epsilon^2$ at most $\Pr[\text{Reach } X_2] \cdot \Pr[X_2 = 1/\epsilon^2] = \epsilon^2$ fraction of the times. Thus its expected value is at most $(1-\epsilon)\cdot 1 + \epsilon^2 \cdot 1/\epsilon^2 = 2 - \epsilon \ll 1/\epsilon$, which is the expected offline optimum.

The last example shows that it does not suffice if we get a few random variables (like X_1) w.p. x_t^* and "miss" the others (like X_2) by getting them only with a small probability, since most of the contribution to the expected offline optimum might be coming from those missed random variables. The idea behind online/offline contention resolution scheme is precisely how to perform the rounding/selection s.t. on average every element gets selected by our algorithm a good fraction of the time.

Contention Resolution Schemes (CRSs) are a powerful tool for offline optimization problems [7]. For a given $\mathbf{x} \in [0,1]^n$, let $R(\mathbf{x})$ denote a random set containing each element $i \in [n]$ independently w.p. x_i. We say an element i is *active* if it belongs to $R(\mathbf{x})$.

Definition 2 (Contention resolution scheme for single-item selection[2])**.** *Given a finite ground set V with $n = |V|$ and an $\mathbf{x} \in [0,1]^n$ satisfying $\sum_t x_t \leq 1$, a c-*selectable *CRS (or simply, c-CRS) is a (randomized) mapping $\pi : 2^V \to 2^V$ satisfying the following three properties:*

(i) $\pi(S) \subseteq S$ *for all* $S \subseteq V$.
(ii) $|\pi(S)| \leq 1$ *for all* $S \subseteq V$.
(iii) $\Pr_{R(\mathbf{x}),\pi}[i \in \pi(R(\mathbf{x}))] \geq c \cdot x_i$ *for all* $i \in V$.

The primary reason why CRS immediately imply a good rounding algorithm is the following observation.

Observation 1. *If f is a monotone linear function then $\mathbb{E}[f(\pi(R(\mathbf{x})))] \geq c \cdot \mathbb{E}[f(R(\mathbf{x}))]$ by linearity of expectation.*

By constructing CRSs for various constraint families of $\mathcal{F}$, Chekuri, Vondrák, and Zenklusen [7] give improved approximation algorithms for linear and submodular maximization problems under knapsack, matroid, matchoid constraints, and their intersections[3].

In the above Definition 2 of offline CRS, the algorithm first flips all the random coins to sample $R(\mathbf{x})$, and then obtains $\pi(R(\mathbf{x})) \subseteq R(\mathbf{x})$. For our prophet inequality problem, this randomness is an inherent part of the problem. Feldman, Svensson, and Zenklusen [5] therefore introduce an OCRS where the random set $R(\mathbf{x})$ is sampled in the same manner, but whether $i \in R(\mathbf{x})$ (or not) is only revealed one-by-one to the algorithm in the order 1 to

[2]The definition can be easily extended to more general packing constraints by defining $P_{\mathcal{F}} \subseteq [0,1]^V$ to be the convex hull of all indicator vectors of feasible sets, working with $\mathbf{x} \in P_{\mathcal{F}}$, and replacing Property (ii) with $\pi(S) \in \mathcal{F}$. E.g., in the case of k-uniform matroids this would mean that $\mathbf{x}$ satisfies $\sum_i x_i \leq k$ and we have to accept at most k elements. For a general matroid this would mean that $\mathbf{x}$ belongs to the matroid polytope and we have to accept an independent set.)

[3]Some "greedy" properties are also required from the CRS for the guarantees to hold for a submodular function f [7].

n. After each revelation (arrival), the OCRS has to irrevocably decide whether to include $i \in R(\mathbf{x})$ into $\pi(R(\mathbf{x}))$ (if possible). A c-selectable OCRS (or simply, c-OCRS) is an OCRS satisfying the above properties (i) to (iii) of a c-CRS.

As a corollary of Lemma 3, the definition of OCRS, and Observation 1 we can prove the following.

Corollary 4. *A c-OCRS for single-item selection implies a $1/c$-competitive PI.*

Proof. We will prove this result only for weighted-Bernoulli distributions but it holds for more general distributions (see HW-3). On arrival of the t-th random variable X_t, we discard it independently with probability $1 - \frac{x_t^*}{p_t}$. Otherwise (i.e., w.p. $\frac{x_t^*}{p_t}$), if $X_t = v_t$ then we send it to the given c-OCRS as an active element and accept it iff the OCRS accepts it. Observe that in this construction the probability with which element i is active equals $\Pr[i \text{ is not discarded}] \cdot \Pr[X_t = v_t] = \frac{x_t^*}{p_t} \cdot p_t = x_t^*$. So, sum of the activation probabilities of all elements $\sum_t x_t^* \leq 1$ and we can apply the c-OCRS. Since the OCRS guarantees that each element is selected w.p. $\geq c \cdot x_i^*$ while being active, we get by linearity of expectation that the expected value of the algorithm $\geq \sum_i c \cdot x_i^* v_i = c \cdot LP^* \geq c \cdot \mathbb{E}[\max_t X_t]$ by Lemma 3. □

In the remaining section we design good OCRS for single-item selection.

A simple $1/4$-OCRS. As a warmup, we discuss a simple 1/4-OCRS for a single-item selection. Given $\mathbf{x}$ satisfying $\sum_i x_i \leq 1$, consider an algorithm that ignores each element i independently w.p. $1/2$, and otherwise selects i only if it is active. Since this algorithm selects any element i w.p. at most $x_i/2$ (when i is not ignored and is active), by Union bound the algorithm selects no element till the end w.p. at least $1 - \sum_i x_i/2 \geq 1/2$. Hence the algorithm reaches each element i w.p. at least $1/2$ without selecting any of the previous elements. Moreover, it does not ignore i w.p. $1/2$, which implies it considers each element w.p. at least $1/4$.

Optimal $1/2$-OCRS. An interesting result of Alaei [6] shows that the above 1/4-OCRS can be improved to a 1/2-OCRS over a rank 1 matroid by "greedily" maximizing the probability of ignoring the next element i, but considering i w.p. $1/2$ on average.

Theorem 5. *There exists a $1/2$-OCRS for single-item selection.*

Since the algorithm selects at most 1 element, the only decision it makes is whether to accept the next element i if it is active. The main idea is to ignore (i.e., not consider) i with maximum probability, while satisfying that on average it is considered at least α fraction of times for some fixed α (we later set $\alpha = 1/2$). Thus on reaching i, the algorithm selects i iff it is both considered and is active.

Proof of Theorem 5. For $i \in [n]$, let r_i denote the probability that the algorithm *reaches* i, i.e., it has not selected any of the elements $[i-1]$. Thus, $r_1 = 1$ and we want $r_n = \alpha$. Let q_i denote the probability that i is considered, conditioned on the event that algorithm reaches i. The algorithm sets q_i s.t. it considers each element w.p. α, i.e.,

$$r_i \cdot q_i = \alpha. \tag{2}$$

Since on reaching i the algorithm accepts it only when it is both considered and active,

$$r_{i+1} = r_i \cdot (1 - q_i x_i).$$

Now using (2), this gives

$$r_{i+1} = r_i - \alpha x_i.$$

Summing over all i and using $r_1 = 1$, we get

$$r_n = r_1 - \alpha \textstyle\sum_i x_i \geq 1 - \alpha.$$

Finally, using $r_n = \alpha$, we get $\alpha \geq 1/2$. □

Optimality of 1/2-OCRS We argue that the factors 1/2 is optimal. This follows from the fact that if there was a better OCRS then we would obtain a better than 2-competitive PI, which contradicts Lemma 1. For a more direct proof, consider $n = 2$, with $x_1 = 1 - \epsilon$ and $x_2 = \epsilon$ for some $\epsilon \to 0$. Since the OCRS algorithm has to select the first element at least 1/2 fraction of the times, it can attempt to select the second element at most $1/2 + \epsilon/2$ fraction of the times.

Notes

Lecture notes partly based on [1, 8].

References

[1] Gupta, Anupam. "Prophets and secretaries". IPCO Tutorial Notes, 2009.

[2] Feldman, Michal and Kesselheim, Thomas and Singla, Sahil, Tutorial on "Prophet Inequalities and Implications to Pricing Mechanisms and Online Algorithms", http://www.thomas-kesselheim.de/tutorial-prophet-inequalities, EC, 2021.

[3] Kleinberg, Robert, and Seth Matthew Weinberg. "Matroid prophet inequalities." Proceedings of the forty-fourth annual ACM Symposium on Theory of computing. 2012.

[4] Rubinstein, Aviad. "Beyond matroids: Secretary problem and prophet inequality with general constraints." Proceedings of the forty-eighth annual ACM Symposium on Theory of Computing. 2016.

[5] Feldman, Moran, Ola Svensson, and Rico Zenklusen. "Online Contention Resolution Schemes with Applications to Bayesian Selection Problems." SIAM Journal on Computing 50.2 (2021): 255-300.

[6] Alaei, Saeed. "Bayesian combinatorial auctions: Expanding single buyer mechanisms to many buyers." SIAM Journal on Computing 43.2 (2014): 930-972.

[7] Chekuri, Chandra, Jan Vondrák, and Rico Zenklusen. "Submodular function maximization via the multilinear relaxation and contention resolution schemes." SIAM Journal on Computing 43.6 (2014): 1831-1879.

[8] Lee, Euiwoong, and Sahil Singla. "Optimal online contention resolution schemes via ex-ante prophet inequalities." arXiv preprint arXiv:1806.09251 (2018).