11
Prophets and Secretaries

The *prophet* and *secretary* problems are two classes of problems where online decision-making meets stochasticity: in the first set of problems the inputs are random variables, whereas in the second one the inputs are worst-case but revealed to the algorithm (a.k.a. the decision-maker) in random order. Here we survey some results, proofs, and techniques, and give some pointers to the rich body of work developing around them.

11.1 The Prophet Problem

The problem setting: there are n random variables $X_1, X_2, \ldots, X_n$. We know their distributions up-front, but not their realizations. These realizations are revealed one-by-one (say in the order $1, 2, \ldots, n$). We want to give a strategy (which is a stopping rule) that, upon seeing the value X_i (and all the values before it) decides either to *choose* i, in which case we get reward X_i and the process stops. Or we can *pass*, in which case we move on to the next items, and are not allowed to come back to i ever again. We want to maximize our expected reward. If

$$X_{\max} := \max\{X_1, X_2, \ldots, X_n\},$$

it is clear that our expected reward cannot exceed $\mathbb{E}[X_{\max}]$. But how close can we get?

If we want to find the best strategy, and we know the order in which we are shown these r.v.s, there is dynamic programming algorithm. (Exercise!)

In fact, we may be off by a factor of almost two against this yardstick in some cases: suppose $X_1 = 1$ surely, whereas $X_2 = 1/\varepsilon$ with probability ε, and 0 otherwise. Any strategy either picks 1 or passes on it, and hence gets expected value 1, whereas $\mathbb{E}[X_{\max}] = (2 - \varepsilon)$. Surprisingly, this is the worst case.

Theorem 11.1 (Krengel, Sucheston, and Garling)**.** *There is a strategy with expected reward* $1/2\mathbb{E}[X_{\max}]$.

Such a result, that gives a stopping rule whose value is comparable to the $\mathbb{E}[X_{\max}]$ is called a *prophet inequality*, the idea being that one can come close to the performance of a prophet who is clairvoyant,

can see the future. The result in Theorem 11.1 was proved by Krengel, Sucheston, and Garling; several proofs have been given since. Apart from being a useful algorithmic construct, the prophet inequality naturally fits into work on algorithmic auction design: suppose you know that n potential are interested in an item with valuations $X_1, \ldots, X_n$, and you want to sell to one person: how do you make sure your revenue is close to $\mathbb{E}[X_{\max}]$?

We now give three proofs of this theorem. For the moment, let us ignore computational considerations, and just talk about the information theoretic issues.

11.1.1 *The Median Algorithm*

This proof is due to Ester Samuel-Cahn.

Let τ be the median of the distribution of $X_{\max}$: i.e.,

$$\Pr[X_{\max} \geq \tau] = 1/2.$$

(For simplicity we assume that there is no point mass at τ, the proof is easily extended to discrete distributions too.) Now the strategy is simple: *pick the first X_i which exceeds τ*. We claim this prove Theorem 11.1.

Proof. Observe that we pick an item with probability exactly $1/2$, but how does the expected reward compare with $\mathbb{E}[X_{\max}]$?

$$\begin{aligned}\mathbb{E}[X_{\max}] &\leq \tau + \mathbb{E}[(X_{\max} - \tau)^+]) \\ &\leq \tau + \mathbb{E}\left[\sum_{i=1}^{n} (X_i - \tau)^+\right].\end{aligned}$$

And what does the algorithm get?

$$\begin{aligned}ALG &\geq \tau \cdot \Pr[X_{\max} \geq \tau] + \sum_{i=1}^{n} \mathbb{E}[(X_i - \tau)^+] \cdot \Pr[\bigwedge_{j \leq i} (X_j < \tau)] \\ &\geq \tau \cdot \Pr[X_{\max} \geq \tau] + \sum_{i=1}^{n} \mathbb{E}[(X_i - \tau)^+] \cdot \Pr[X_{\max} < \tau]\end{aligned}$$

But both these probability values equal half, and hence $ALG \geq 1/2\, \mathbb{E}[X_{\max}]$. □

While a beautiful proof, it is somewhat mysterious, and difficult to generalize. Indeed, suppose we are allowed to choose up to k variables to maximize the sum of their realizations? The above proof seems difficult to generalize, but the following LP-based one will.

However, a recent paper of Shuchi Chawla, Nikhil Devanur, and Thodoris Lykouris gives an extension of Samuel-Cahn's proof to the multiple item setting.

11.1.2 *The LP-based Algorithm*

The second proof is due to Shuchi Chawla, Jason Hartline, David Malec, and Balu Sivan, and to Saeed Alaei. Define p_i as the proba-

If you know of an earlier reference for this proof, please let me know.

bility that element X_i takes on the highest value. Hence $\sum_i p_i = 1$. Moreover, suppose τ_i is such that $\Pr[X_i \geq \tau_i] = p_i$, i.e., the p_i^{th} percentile for X_i. Define

$$v_i(p_i) := \mathbb{E}[X_i \mid X_i \geq \tau_i]$$

as the value of X_i conditioned on it lying in the top p_i^{th} percentile. Clearly, $\mathbb{E}[X_{\max}] \leq \sum_i v_i(p_i) \cdot p_i$. Here's an algorithm that gets value $1/4 \sum_i v_i(p_i) \cdot p_i \geq 1/4\, \mathbb{E}[X_{\max}]$:

> *If we have not chosen an item among $1, \ldots, i-1$, when looking at item i, pass with probability half without even looking at X_i, else accept it if $X_i \geq \tau_i$.*

Lemma 11.2. *The algorithm achieves a value of $1/4\mathbb{E}[X_{\max}]$.*

Proof. Say we "reach" item i if we've not picked an item before i. The expected value of the algorithm is

$$\begin{aligned} ALG &\geq \sum_{i=1}^{n} \Pr[\text{reach item } i] \cdot 1/2 \cdot \Pr[X_i \geq \tau_i] \cdot \mathbb{E}[X_i \mid X_i \geq \tau_i] \\ &= \sum_{i=1}^{n} \Pr[\text{reach item } i] \cdot 1/2 \cdot p_i \cdot v_i(p_i). \end{aligned} \tag{11.1}$$

Since we pick each item with probability $p_i/2$, the expected number of items we choose is half. So, by Markov's inequality, the probability we pick no item at all is at least half. Hence, the probability we reach item i is at least one half too, the above expression is $1/4 \sum_i v_i(p_i) \cdot p_i$ as claimed. □

Now to give a better bound, we refine the algorithm above: suppose we denote the probability of reaching item i by r_i, and suppose we reject item i outright with probability $1 - q_i$. Then (11.1) really shows that

$$ALG \geq \sum_{i=1}^{n} r_i \cdot q_i \cdot p_i \cdot v_i(p_i).$$

Above, we ensured that $q_i = r_i = 1/2$, and hence lost a factor of $1/4$. But if we could ensure that $r_i \cdot q_i = 1/2$, we'd get the desired result of $1/2\mathbb{E}[X_{\max}]$. For the first item $r_1 = 1$ and hence we can set $q_1 = 1/2$. What about future items? Note that since that

$$r_{i+1} = r_i(1 - q_i \cdot p_i) \tag{11.2}$$

we can just set $q_{i+1} = \frac{1}{2r_{i+1}}$. A simple induction shows that $r_{i+1} \geq 1/2$—indeed, sum up (11.2) to get $r_{i+1} = r_1 - \sum_{j \leq i} p_i/2$—so that $q_{i+1} \in [0, 1]$ and is well-defined.

11.1.3 The Computational Aspect

If the distribution of the r.v.s X_i is explicitly given, the proofs above immediately give us algorithms. However, what if the variables X_i are given via a black-box that we can only access via samples?

The first proof merely relies on finding the median: in fact, finding an "approximate median" $\widehat{\tau}$ such that $\Pr[X_{\max} < \widehat{\tau}] \in (1/2 - \varepsilon, 1/2 + \varepsilon)$ gives us expected reward $1/2+\varepsilon/2\mathbb{E}[X_{\max}]$. To do this, sample from $X_{\max}$ $O(\varepsilon^{-2} \log \delta^{-1})$ times (each sample to $X_{\max}$ requires one sample to each of the X_is) and take $\widehat{\tau}$ to be the sample median. A Hoeffding bound shows that $\widehat{\tau}$ is an "approximate median" with probability at least $1 - \delta$.

For the second proof, there are two ways of making it algorithmic. Firstly, if the quantities are polynomially bounded, estimate p_i and v_i by sampling. Alternatively, solve the convex program

$$\max \left\{ \sum_i y_i \cdot v_i(y_i) \mid \sum_i y_i = 1 \right\}$$

and use the y_i's from its solution in lieu of p_i's in the algorithm above.

Do we need such good approximations? Indeed, getting samples from the distributions may be expensive, so how few samples can we get away with? A paper of Pablo Azar, Bobby Kleinberg, and Matt Weinberg shows how to get a constant fraction of $\mathbb{E}[X_{\max}]$ via taking just one sample from each of the X_is. Let us a give a different algorithm, by Aviad Rubinstein, Jack Wang, and Matt Weinberg.

11.1.4 The One-Sample Algorithm

For the preprocessing, take one sample from the distributions for each of $X_1, X_2, \ldots, X_n$. (Call these samples $S_1, S_2, \ldots, S_n$.) Set the threshold τ to be the largest of these sample values. Now when seeing the actual items $X_1, X_2, \ldots, X_n$, pick the first item higher than τ. We claim this one-sample algorithm gives an expected value at least $1/2\,\mathbb{E}[X_{\max}]$.

Proof. As a thought experiment, consider taking two independent samples from each distribution, then flipping a coin C_i to decide which is X_i and which is S_i. This has the same distribution as original process, so we consider this perspective.

Now consider all these $2n$ values together, in sorted order: call these $W_1 \geq W_2 \geq \ldots \geq W_{2n}$. We say W_j has *index* i if it is drawn from the i^{th} distribution, and hence equal to X_i or S_i. Let j^* be the smallest position where W_i, W_{j^*+1} have the same index for some $i \leq j^*$. Observe: the coins C_i for the indices corresponding to the first

j^* positions are independent, and the coin for the position j^*+1 is the same as one of the previous ones. We claim that

$$\mathbb{E}[X_{\max}] = \sum_{i \le j^*} \frac{W_i}{2^i} + \frac{W_{j^*+1}}{2^{j^*}}.$$

Indeed, $X_{\max} = W_i$ if all the previous $W_{i'}$s belong to the sample (i.e., they are S's and not X's), but W_i belongs to the actual values (it is an X). Moreover, if all the previous values are Ss, then W_{j^*+1} would be an X and hence the maximum, by our choice of j^*.

What about the algorithm? If W_1 is a sample (i.e., an S-value) then we don't get any value. Else if $W_1, \ldots, W_i$ are all X values, and W_{i+1} is a sample (S value) then we get value at least W_i. If $i < j^*$, this happens with probability $\frac{1}{2^{i+1}}$ since all the $i+1$ coins are independent. Else if $i = j^*$, the probability is $\frac{1}{2^i} = \frac{1}{2^{j^*}}$. Hence

$$\text{Alg} \ge \sum_{i<j^*} \frac{W_i}{2^{i+1}} + \frac{W_{j^*}}{2^{j^*}} \ge \sum_{i<j^*} \frac{W_i}{2^{i+1}} + \frac{W_{j^*}}{2^{j^*+1}} + \frac{W_{j^*+1}}{2^{j^*+1}}.$$

But this is at least half of $\mathbb{E}[X_{\max}]$, which proves the theorem. □

11.1.5 *Extensions: Picking Multiple Items*

What about the case where we are allowed to choose k variables from among the n? Proof #2 generalizes quite seamlessly. If p_i is the probability that X_i is among the top k values, we now have:

$$\textstyle\sum_i p_i = k. \tag{11.3}$$

The "upper bound" on our quantity of interest remains essentially unchanged:

$$\mathbb{E}[\text{sum of top } k \text{ r.v.s}] \le \textstyle\sum_i v_i(p_i) \cdot p_i. \tag{11.4}$$

What about an algorithm to get value $1/4$ of the value in (11.4)? The same as above: reject each item outright with probability $1/2$, else pick i if $X_i \ge \tau_i$. Proof #2 goes through unchanged.

For this case of picking multiple items, we can do much better: a result of Alaei shows that one can get within $(1 - 1/\sqrt{k+3})$ of the value in (11.4)—for $k = 1$, this matches the factor $1/2$ we showed above. One can, however, get a factor of $(1 - O(\sqrt{\frac{\log k}{k}}))$ using a simple concentration bound.

Proof. Suppose we reduce the rejection probability to δ. Then the probability that we reach some item i without having picked k items already is lower-bounded by the probability that we pick at most k elements in the entire process. Since we reject items with probability

δ, the expected number of elements we pick is $(1-\delta)k$. Hence, the probability that we pick less than k items is at least $1 - e^{-\Theta(\delta^2 k)}$, by a Hoeffding bound for sums of independent random variables. Now setting $\delta = O(\sqrt{\frac{\log k}{k}})$ ensures that the probability of reaching each item is at least $(1 - \frac{1}{k})$, and an argument similar to that in Proof #2 shows that

$$\begin{aligned} ALG &\geq \textstyle\sum_{i=1}^{n} \Pr[\text{reach item } i] \cdot \Pr[\text{not reject item } i] \cdot \Pr[X_i \geq \tau_i] \cdot \mathbb{E}[X_i \mid X_i \geq \tau_i] \\ &= \textstyle\sum_{i=1}^{n} (1 - 1/k) \cdot (1 - O(\sqrt{\frac{\log k}{k}})) \cdot p_i \cdot v_i(p_i), \end{aligned}$$

which gives the claimed bound of $(1 - O(\sqrt{\frac{\log k}{k}}))$. □

11.1.6 *Extensions: Matroid Constraints*

Suppose there is a matroid structure $\mathcal{M}$ with ground set $[n]$, and the set of random variables we choose must be independent in this matroid $\mathcal{M}$. The value of the set is the sum of the values of items within it. (Hence, the case of at most k items above corresponds to the *uniform* matroid of rank k.) The goal is to make the expected value of the set picked by the algorithm close to the expected value of the max-weight independent set.

Recall that a ***matroid*** $\mathcal{M} = (U, \mathcal{F})$ is a set U is a collection of subsets $\mathcal{F} \subseteq 2^U$ that is closed under taking subsets, such that if $A, B \in \mathcal{F}$ and $|A| < |B|$ then there exists $b \in B \setminus A$ such that $A \cup \{b\} \in \mathcal{F}$. Sets in $\mathcal{F}$ are called *independent sets*.

Bobby Kleinberg and Matt Weinberg give an algorithm to picks an independent set whose expected value is at least half the value of the max-weight independent set, thereby extending the original single-item prophet inequality seamlessly to all matroids. While their original proof uses a combinatorial idea, a LP-based proof was subsequently given by Moran Feldman, Ola Svensson, and Rico Zenklusen. The idea is again clever and conceptually clean: find a solution y to the convex program

$$\begin{aligned} &\textstyle\sum_i v_i(y_i) \cdot y_i. \\ &y \in \text{the matroid polytope for } \mathcal{M} \end{aligned}$$

Now given a fractional point y in the matroid polytope, how to get an integer point (i.e., an independent set). For this they give an approach called an "online contention resolution" scheme that ensures that any item i is picked with probability at least $\Omega(y_i)$, much like in the single-item and k-item cases.

There are many other extensions to prophet inequalities: people have studied more general constraint sets, submodular valuations instead of just additive valuations, what if the order of items is not known, what if we are allowed to choose the order, etc. See papers on arXiv, or in the latest conferences for much more.

11.1.7 Exercises

1. Give a dynamic programming algorithm for the best strategy when we know the order in which r.v.s are revealed to us. (Footnote 1). Extend this to the case where you can pick k items.

   Open problem: is this "best strategy" problem computationally hard when we are given a general matroid constraint? Even a laminar matroid or graphical matroid?

2. If we can choose the order in which we see the items, show that we can get expected value $\geq (1 - 1/e)\mathbb{E}[X_{\max}]$. (Hint: use proof #2, but consider the elements in decreasing order of $v_i(p_i)$.)

   Open problem: can you beat $(1 - 1/e)\mathbb{E}[X_{\max}]$? A recent paper of Abolhassani et al. does so for i.i.d. X_is.

11.2 Secretary Problems

The problem setting: there are n items, each having some intrinsic non-negative value. For simplicity, assume the values are distinct, but we know nothing about their ranges. We know n, and nothing else. The items are presented to us one-by-one. Upon seeing an item, we can either pick it (in which case the process ends) or we can pass (but then this item is rejected and we cannot ever pick it again). The goal is to maximize the probability of picking the item with the largest value $v_{\max}$.

If an adversary chooses the order in which the items are presented, every deterministic strategy must fail. Suppose there are just two items, the first one with value 1. If the algorithm picks it, the adversary can send a second item with value 2, else it sends one with value $1/2$. Randomizing our algorithm can help, but we cannot do much better than $1/n$.

So the secretary problem asks: *what if the items are presented in uniformly random order?* For this setting, it seems somewhat surprising at first glance that one can pick the best item with probability at least a constant (knowing nothing other than n, and the promise of a uniformly random order). Indeed, here a simple algorithm and proof showing a probability of $1/4$:

> Ignore the first $n/2$ items, and then pick the next item that is better than all the ones seen so far.

Note that this algorithm succeeds if the best item is in the second half of the items (which happens w.p. $1/2$) and the second-best item is in the first half (which, conditioned on the above event, happens w.p. $\geq 1/2$). Hence $1/4$. It turns out that rejecting the first half of the items is not optimal, and there are other cases where the algorithm succeeds that this simple analysis does not account for, so let's be more careful. Consider the following *37%-algorithm*:

> Ignore the first n/e items, and then pick the next item that is better than all the ones seen so far.

Theorem 11.3. *As $n \to \infty$, the 37%-algorithm picks the highest number with probability at least $1/e$. Hence, it gets expected value at least $v_{\max}/e$. Moreover, n/e is the optimal choice of m among all wait-and-pick algorithms.*

Proof. Call a number a *prefix-maximum* if it is the largest among the numbers revealed before it. Notice being the maximum is a property of just the set of numbers, whereas being a prefix-maximum is a property of the random sequence and the current position. If we pick the first prefix-maximum after rejecting the first m numbers, the probability we pick the maximum is

$$\sum_{t=m+1}^{n} \Pr[v_t \text{ is max}] \cdot \Pr[\text{max among first } t-1 \text{ numbers falls in first } m \text{ positions}]$$
$$\overset{(\star)}{=} \sum_{t=m+1}^{n} \frac{1}{n} \cdot \frac{m}{t-1} \quad = \quad \frac{m}{n}\big(H_{n-1} - H_{m-1}\big),$$

where $H_k = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k}$ is the k^{th} harmonic number. The equality $(\star)$ uses the uniform random order. Now using the approximation $H_k \approx \ln k + 0.57$ for large k, we get the probability of picking the maximum is about $\frac{m}{n} \ln \frac{n-1}{m-1}$ when m, n are large. This quantity has a maximum value of $1/e$ if we choose $m = n/e$. □

Next we show we can replace any strategy (in a comparison-based model) with a wait-and-pick strategy without decreasing the probability of picking the maximum.

Theorem 11.4. *The strategy that maximizes the probability of picking the highest number can be assumed to be a wait-and-pick strategy.*

Proof. Think of yourself as a player trying to maximize the probability of picking the maximum number. Clearly, you should reject the next number v_i if it is not prefix-maximum. Otherwise, you should pick v_i only if it is prefix-maximum and the probability of v_i being the maximum is more than the probability of you picking the maximum in the remaining sequence. Let us calculate these probabilities.

We use Pmax to abbreviate "prefix-maximum". For position $i \in \{1, \ldots, n\}$, define

$$f(i) = \Pr[v_i \text{ is max} \mid v_i \text{ is Pmax}] \overset{(\star)}{=} \frac{\Pr[v_i \text{ is max}]}{\Pr[v_i \text{ is Pmax}]} \overset{(\star\star)}{=} \frac{1/n}{1/i} = \frac{i}{n},$$

where equality $(\star)$ uses that the maximum is also a prefix-maximum, and $(\star\star)$ uses the uniform random ordering. Note that $f(i)$ increases with i.

Now consider a problem where the numbers are again being revealed in a random order but we must reject the first i numbers. The

goal is to still maximize the probability of picking the highest of the n numbers. Let $g(i)$ denote the probability that the optimal strategy for this problem picks the global maximum.

The function $g(i)$ must be a non-increasing function of i, else we could just ignore the $(i+1)^{st}$ number and set $g(i)$ to mimic the strategy for $g(i+1)$. Moreover, $f(i)$ is increasing. So from the discussion above, you should not pick a prefix-maximum number at any position i where $f(i) < g(i)$ since you can do better on the suffix. Moreover, when $f(i) \geq g(i)$, you should pick v_i if it is prefix-maximum, since it is worse to wait. Therefore, the approach of waiting until f becomes greater than g and thereafter picking the first prefix-maximum is an optimal strategy. □

In keeping with the theme of this chapter, we now give an alternate proof that uses a convex-programming view of the process. We will write down an LP that captures some properties of any feasible solution, optimize this LP and show a strategy whose success probability is comparable to the objective of this LP! The advantage of this approach is that it then extends to adding other constraints to the problem.

Proof. (Due to Niv Buchbinder, Kamal Jain, and Mohit Singh.) Let us fix an optimal strategy. By the first proof above, we know what it is, but let us ignore that for the time being. Let us just assume w.l.o.g. that it does not pick any item that is not the best so far (since such an item cannot be the global best).

Let p_i be the probability that this strategy picks an item at position i. Let q_i be the probability that we pick an item at position i, *conditioned on it being the best so far*. So $q_i = \frac{p_i}{1/i} = i \cdot p_i$.

Now, the probability of picking the best item is

$$\begin{aligned}&\sum_i \Pr[i^{th} \text{ position is global best and we pick it }] \\ &= \sum_i \Pr[i^{th} \text{ position is global best }] \cdot q_i = \sum_i \frac{1}{n} q_i = \sum_i \frac{i}{n} p_i. \qquad (11.5)\end{aligned}$$

What are the constraints? Clearly $p_i \in [0,1]$. But also

$$\begin{aligned}p_i &= \Pr[\text{ pick item } i \mid i \text{ best so far}] \cdot \Pr[i \text{ best so far}] \\ &\leq \Pr[\text{ did not pick } 1,\ldots,i-1 \mid i \text{ best so far}] \cdot (1/i) \qquad (11.6)\end{aligned}$$

But not picking the first $i-1$ items is independent of i being the best so far, so we get

$$p_i \leq \frac{1}{i}\Big(1 - \sum_{j<i} p_j\Big).$$

Hence, the success probability of any strategy (and hence of the optimal strategy) is upper-bounded by the following LP in variables p_i:

$$\begin{aligned} \max \sum_i \frac{i}{n} \cdot p_i & \\ i \cdot p_i &\leq 1 - \sum_{j<i} p_j \\ p_i &\in [0,1]. \end{aligned}$$

Now it can be checked that the solution $p_i = 0$ for $i \leq \tau$ and $p_i \frac{\tau}{n}(\frac{1}{i-1} - \frac{1}{i})$ for $\tau \leq i \leq n$ is a feasible solution, where τ is defined by the smallest value such that $H_{n-1} - H_{\tau-1} \leq 1$. (By duality, we can also show it is optimal!)

Finally we can get a stopping strategy whose success probability matches that of the LP. Indeed, solve the LP. Now, for the i^{th} position if we've not picked an item already and if this item is the best so far, pick it with probability $\frac{ip_i}{1-\sum_{j<i} p_j}$. By the LP constraint, this probability $\in [0,1]$. Moreover, removing the conditioning shows we pick an item at location i with probability p_i, and a calculation similar to the one above shows that our algorithm's success probability is $\sum_i ip_i/n$, the same as the LP. □

11.2.1 *Extension: Game-Theoretic Issues*

Note that in the optimal strategy, we don't pick any items in the first n/e timesteps, and then we pick items with quite varying probabilities. If the items are people interviewing for a job, this gives them an incentive to not come early in the order. Suppose we insist that for each position i, the probability of picking the item at position i is the same. What can we do then?

Let's fix any such strategy, and write an LP capturing the success probabilities of this strategy with uniformity condition as a constraint. Suppose $p \leq 1/n$ is this uniform probability (over the randomness of the input sequence). Again, let q_i be the probability of picking an item at position i, conditioned on it being the best so far. Note that we may pick items even if they are not the best so far, just to satisfy the uniformity condition; hence instead of $q_i = i \cdot p$ as before, we have

$$q_i \leq ip.$$

Moreover, by the same argument as (11.6), we know that

$$q_i \leq 1 - (i-1)p.$$

And the strategy's success probability is again $\sum q_i/n$ using (11.5). So

we can now solve the LP

$$\max \sum_i \frac{1}{n} \cdot q_i$$
$$q_i \le 1 - (i-1) \cdot p$$
$$q_i \le i \cdot p$$
$$q_i \in [0,1], p \ge 0$$

Now the Buchbinder, Jain, and Singh paper shows the optimal value of this LP is at least $1 - 1/\sqrt{2} \approx 0.29$; they also give a slightly more involved algorithm that achieves this success probability.

11.2.2 *Extension: Multiple Items*

Now back to having no restrictions on the item values. Suppose we want to pick k items, and want to maximize the expected sum of these k values. Suppose the set of the k largest values is $S^\star \subseteq [n]$, and their total value is $V^\star = \sum_{i \in S} v_i$. It is easy to get an algorithm with expected value $\Omega(V^\star)$. E.g., split the n items into k groups of n/k items, and run the single-item algorithm separately on each of these. (Why?) Or ignore the first half of the elements, look at the value $\hat{v}$ of the $(1-\varepsilon)k/2^{th}$ highest value item in this set, and pick all items in the second half with values greater than $\hat{v}$. And indeed, ignoring half the items must lose a constant factor in expectation.

But here's an algorithm that gives value $V^\star(1-\delta)$ where $\delta \to 0$ as $k \to \infty$. We will set $\delta = O(k^{-1/3} \log k)$ and $\varepsilon = \delta/2$. Ignore the first δn items. (We expect $\delta k \approx k^{2/3}$ items from $S^\star$ fall in this ignored set.) Now look at the value $\hat{v}$ of the $(1-\varepsilon)\delta k^{th}$-highest valued item in this ignored set, and pick the first (at most) k elements with values greater than $\hat{v}$ along the remaining $(1-\delta)n$ elements.

Why is this algorithm good? There are two failure modes: (i) if $v' = \min_{i \in S^\star} v_i$ be the lowest value item we care about, then we don't want $\hat{v} \le v'$ else we may pick low valued items, and (ii) we want the number of items from $S^\star$ in the last $(1-\delta)n$ and greater than $\hat{v}$ to be close to k.

Let's sketch why both bad events happen rarely. For event (i) to happen, fewer than $(1-\varepsilon)\delta k$ items from $S^\star$ fall in the first δn locations: i.e., their number is less than $(1-\varepsilon)$ times its expectation, which has probability $\exp(-\varepsilon^2 \delta k) = 1/\operatorname{poly}(k)$ by a Hoeffding bound. For event (ii) to happen, more than $(1-\varepsilon)\delta k$ of the top $(1-\delta)k$ items from $S^\star$ fall among the ignored items. This means their number exceeds $(1+O(\varepsilon))$ times its expectation, which again has probability $\exp(-\varepsilon^2 \delta k) = 1/\operatorname{poly}(k)$.

An aside: the standard concentration bounds we know are for sums of i.i.d. r.v.s whereas the random order model causes correlations. The easiest way to handle that is to ignore not the first δn items but a random number of items $\sim \operatorname{Bin}(n, \delta)$. Then each item has probability δ of being ignored, independent of others.

Is this tradeoff optimal? No. Kleinberg showed that one can get expected value $V^\star(1 - O(k^{-1/2}))$, and this is asymptotically optimal.

In fact, one can extend this even further: a set of vectors $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n \in [0,1]^m$ is fixed, along with values $v_1, v_2, \ldots, v_n$. These are initially unknown. Now they are revealed one-by-one to the algorithm in a uniformly random order. The algorithm, on seeing a vector and its value must decide to pick it or irrevocably reject it. It can pick as many vectors as it wants, subject to their sum being at most k in each coordinate; the goal is to maximize the expected value of the picked vectors. The k-secretary case is the 1-dimensional case when each $\mathbf{a}_i = (1)$. Indeed, this is the problem of solving a packing linear program online, where the columns arrive in random order. A series of works have extended the k-secretary case to this online packing LP problem, getting values which are $(1 - O(\sqrt{(\log m)/k}))$ times the optimal value of the LP.

11.2.3 *Extension: Matroids*

One of the most tantalizing generalizations of the secretary problem is to matroids. Suppose the n elements form the ground set of a matroid, and the elements we pick must form an independent set in this matroid. Babioff, Immorlica, and Kleinberg asked: if the max-weight independent set has value V^*, can we get $\Omega(V^*)$ using an online algorithm? The current best algorithms, due to Lachish, and to Feldman, Svensson, and Zenklusen, achieve expected value $\Omega(V^*/\log\log k)$, where k is the rank of the matroid. Can we improve this further, say to a constant? A constant factor is known for many classes of matroids, like graphical matroids, laminar matroids, transversal matroids, and gammoids.

11.2.4 *Other Random Arrival Models*

One can consider other models for items arriving online: say a set of n items (and their values) is fixed by an adversary, and each timestep we see one of these items sampled uniformly *with replacement*. (The random order model is same, but without replacement.) This model, called the *i.i.d. model*, has been studied extensively—results in this model are often easier than in the random order model (due to lack of correlations). See, e.g., references in a monograph by Aranyak Mehta.

Do we need the order of items to be uniformly random, or would weaker assumptions suffice? Kesselhiem, Kleinberg, and Niazadeh consider this question in a very nice paper and show that much less independence is enough for many of these results to hold [1].

[1]

In general the random-order model is a clean way of modeling the fact that an online stream of data may not be adversarially ordered. Many papers in online algorithms have used this model to give better

results than in the worst-case model: some of my favorite ones are paper of Meyerson on facility location, and this paper of Bahmani, Chowdhury, and Goel on computing PageRank incrementally.

Again, see online for many many papers related to the secretary problem: numerous models, different constraints on what sets of items you can pick, and how you measure the quality of the picked set. It's a very clean model, and can be used in many different settings.

Exercises

1. Give an algorithm for general matroids that finds an independent set with expected value at least an $O(1/(\log k))$-fraction of the max-value independent set.
2. Improve the above result to $O(1)$-fraction for graphic matroids.

11.3 Multi-Armed Bandits

We now switch to the "Stochastic Multi-Armed Bandit" (MAB) problem and just sketch two approaches in that setting. There are K arms (slot machines). Arm i gives a reward of 1 with probability p_i (unknown) and 0 otherwise. We want a strategy to play arms over T time steps to maximize total reward. We define **Regret** as the difference between the expected reward of the optimal arm and the expected reward collected by our algorithm:

$$\text{Regret} = T \cdot p_{\max} - E\left[\sum_{t=1}^{T} \text{Reward}_t\right] = \sum_{j=1}^{K} \Delta_j E[N_j],$$

where $p_{\max} = \max_i p_i$, $\Delta_j = p_{\max} - p_j$, and N_j is the number of times arm j is pulled.

Theorem 11.5. *There exist strategies with Regret* $\leq O(\sqrt{KT \log T})$.

We will look at two algorithms: Active Arm Elimination and UCB.

11.3.1 Active Arm Elimination (AAE)

This algorithm operates in phases. Initialize active set $S = \{1, \ldots, K\}$. In phase $r = 1, 2, \ldots$:

1. Set $\epsilon_r = 2^{-r}$.
2. Play each arm in S for $O(\frac{1}{\epsilon_r^2} \log T)$ times.
3. Let $\hat{p}_j$ be the empirical mean of arm j.
4. Eliminate arm j from S if $\hat{p}_j < (\max_{k \in S} \hat{p}_k) - 2\epsilon_r$.

Analysis: Using Chernoff bounds, with high probability, $\hat{p}_j \in p_j \pm \epsilon_r$. This ensures that the best arm is never eliminated, and any arm j with $\Delta_j > 4\epsilon_r$ is eliminated. An arm j is played until phase r such that $\epsilon_r \approx \Delta_j/4$. The number of pulls N_j is roughly:

$$N_j \approx \sum_{r:2^{-r} \geq \Delta_j} \frac{\log T}{(2^{-r})^2} = O\left(\frac{\log T}{\Delta_j^2}\right) + 1.$$

The total regret is:

$$\mathbb{E}[\text{Regret}] = \sum_{j:\Delta_j>0} N_j \Delta_j = \sum_j O\left(\frac{\log T}{\Delta_j}\right) + \Delta_j.$$

If Δ_j's are small we can get a different bound as stated in Theorem 11.5: Regret $\leq O(\sqrt{KT \log T})$. The proof is

$$\begin{aligned}
\sum_j \Delta_j \mathbb{E}[N_j] &= \sum_j \sqrt{\Delta_j^2 \mathbb{E}[N_j]} \sqrt{\mathbb{E}[N_j]} \\
&\leq \sqrt{\sum \Delta_j^2 \mathbb{E}[N_j]} \sqrt{\sum \mathbb{E}[N_j]} \qquad \text{(Cauchy-Schwarz)} \\
&= O(\sqrt{KT \log T}).
\end{aligned}$$

11.3.2 *Upper Confidence Bound (UCB)*

The UCB algorithm follows the principle of "optimism in the face of uncertainty". Let $N_j(t)$ be the number of times arm j has been pulled up to time t, and $\hat{p}_j(t)$ be the empirical average reward. We define the upper confidence bound (or index) for arm j at time t:

$$\tilde{p}_j(t) = \hat{p}_j(t) + \sqrt{\frac{c \log(TK)}{N_j(t)}},$$

where c is a constant (to make concentration bounds work).

Algorithm: At each step t, pick the arm j that maximizes $\tilde{p}_j(t)$. Update N_j and $\hat{p}_j$.

Analysis: We analyze the regret by conditioning on a "good event" where all empirical estimates are close to their true means. Assume w.l.o.g. that arm 1 is optimal ($p_1 > p_2 \geq \cdots \geq p_K$).

Fact 11.6. With probability at least $1 - \frac{1}{T^2}$, the empirical means are within the confidence intervals for all times t and all arms j. Specifically,

$$\Pr\left(\hat{p}_j(t) \in p_j \pm \sqrt{\frac{c \log(TK)}{N_j(t)}} \quad \forall t, j\right) \geq 1 - \frac{1}{T^2}.$$

We call this the "Good Event" $\mathcal{E}$.

Fact 11.7. If the "Good Event" $\mathcal{E}$ holds, then for all arms j:

$$\tilde{p}_j(t) \geq p_j.$$

In particular, for the optimal arm (arm 1), $\tilde{p}_1(t) \geq p_1$.

Proof. Under $\mathcal{E}$, $\hat{p}_j \geq p_j - \sqrt{\frac{c\log(TK)}{N_j}}$. Thus $\tilde{p}_j = \hat{p}_j + \sqrt{\frac{c\log(TK)}{N_j}} \geq p_j$. □

Fact 11.8. If a suboptimal arm j is chosen at time t (i.e., $j = \text{argmax}_k \tilde{p}_k(t)$) and $\mathcal{E}$ holds, then:

$$\tilde{p}_j(t) \geq \tilde{p}_1(t) \geq p_1.$$

Using these facts, we can bound the number of pulls for a suboptimal arm j. If j is pulled under $\mathcal{E}$, then:

$$\hat{p}_j(t) + \sqrt{\frac{c\log(TK)}{N_j(t)}} \geq p_1.$$

Since $\mathcal{E}$ also implies $\hat{p}_j(t) \leq p_j + \sqrt{\frac{c\log(TK)}{N_j(t)}}$, we substitute this in:

$$\left(p_j + \sqrt{\frac{c\log(TK)}{N_j(t)}}\right) + \sqrt{\frac{c\log(TK)}{N_j(t)}} \geq p_1$$

$$p_j + 2\sqrt{\frac{c\log(TK)}{N_j(t)}} \geq p_1 = p_j + \Delta_j.$$

This simplifies to:

$$2\sqrt{\frac{c\log(TK)}{N_j(t)}} \geq \Delta_j \implies N_j(t) \leq \frac{O(\log(TK))}{\Delta_j^2} \quad \text{in this case.}$$

Regret Bound. Of course, with probability $1/T^2$ the good event fails, and N_j could be up to T.

$$E[N_j] \leq 1 + \frac{O(\log(KT))}{\Delta_j^2}.$$

(The constant 1 comes from the bad event $T \cdot \frac{1}{T^2} \leq 1$ and/or the initial pull).

The total expected regret is:

$$\text{Regret} = \sum_{j=2}^{K} \Delta_j E[N_j] \leq \sum_{j=2}^{K} \left(\Delta_j + \frac{O(\log(KT))}{\Delta_j}\right).$$

Using the same Cauchy-Schwarz argument as in AAE, we obtain the bound:

$$\text{Regret} \leq O(\sqrt{KT\log(KT)}).$$

(Can probably remove the K from inside the log term, but small improvement).