Brian Chen 14 Dec 2016

Multiple-Choice Prophet Inequalities

6.854, as taught by D. Karger, A. Mądry

Abstract

The well-established Prophet Inequality concerns an online algorithm that observes a sequence of random variables $X_1, \ldots, X_n$ with known distribution in order, and must decide, when it observes the value w_i of each variable X_i, whether to end the game by accepting w_i and obtain reward $R = w_i$, or to reject w_i and continue with no chance of returning. The algorithm is compared to a "prophet" that can see all the w_i ahead of time and simply pick the best one. The optimal competitive ratio has been known to be 2 for this problem since 1976. We survey some results related to generalizations of the Prophet Inequality where the prophet and algorithm are allowed to choose multiple variables and obtain reward equal to their sum.

1 Introduction

The "Prophet Inequality" is an important result in optimal stopping theory, a body of work concerned with online algorithms for choosing when to take an action with the goal of maximizing some reward. The foundational Prophet Inequality was proven in 1978 by Krengel and Sucheston [9].

In many regards, the Prophet Inequality is similar to the famous Secretary Problem, possibly the most famous problem in optimal stopping theory, in which an online algorithm observes a series of candidates and tries to stop at the best candidate or maximize the expected value of the chosen candidate, likewise with no chance of returning to previous candidates after they have been rejected. However, in the Secretary Problem the order of candidates is randomized while the values and their distributions are adversarial and unknown, while in the Prophet Inequality the order of candidates is fixed adversarially but known and the values are randomly drawn from known distributions. Despite these differences, the problems are tightly interrelated; for example, Azar, Kleinberg, and Weinberg [3] develop a method for converting many secretary problem algorithms into prophet inequality algorithms.

The Prophet Inequality and related results have found widespread applications in the field of designing *Bayesian optimal mechanisms*, which are roughly mechanisms for maximizing profit from selling items to potential buyers. The original Prophet Inequality can be applied as a way to sell one item to one of a list of potential buyers, who will name prices drawn from known distributions not known ahead of time. The idea of studying generalizations of the prophet inequality where the algorithm can accept multiple values arises naturally when one has multiple items for sale. Different constraints on the prophet inequality problem arise with different bidding market rules.

Many generalizations and modifications of the Prophet Inequality have been studied (see, for instance, Hill and Kertz [7] for an earlier survey). In this survey, we will primarily focus on Prophet Inequalities where multiple values can be accepted instead of only one. We will look at increasingly general restrictions on what subsets of values may be accepted; the allowed sets are termed "feasible sets". We will discuss several choices for constraints that might be imposed on the families of feasible sets, primarily "subsets of at most k values", "subsets belonging to a *matroid*", and "subsets belonging to an arbitrary downward-closed family". We will examine shared concepts and techniques, including the idea of thresholds.

2 Preliminaries

In this section, we define some basic terms relating to the prophet inequality.

Definition 1. *An instance of the* **prophet inequality problem** *consists of a sequence of nonnegative random variables $X_1, X_2, \ldots, X_n$, which must be indepdendent but are not necessarily identically distributed.*

Note that, for our purposes, the order of these variables shall be fixed and known ahead of time. Some authors also analyze the problem where the order can be chosen by the algorithm, or adversarially and dynamically chosen based on the algorithm's choices and revealed values.

Definition 2. *An* **online algorithm** *for the* PROPHET *inequality problem is an algorithm that knows the distributions of $X_1, X_2, \ldots, X_n$ beforehand, but receives the actual values w_i of X_i online one at a time. Right after the algorithm receives the value w_i of some X_i, it must decide irrevocably whether to* accept *X_i, in which case the problem ends and the algorithm receives reward $R = w_i$, or* reject *it, in which case the problem continues with X_{i+1} and following random variables. (We will somewhat interchangeably say that the algorithm accepts a variable or that it accepts a value.) The algorithm's goal is to maximize its reward R.*

In comparison, a **"prophet"**[1] *for the prophet inequality problem similarly has to accept one of the X_i, but can see all of the actual values w_i of X_i before deciding which w_i to accept. The prophet also receives award equal to the value it accepts, and also seeks to maximize that reward. Clearly, the prophet will just pick the maximum w_i. We will denote the reward of the prophet (a random variable) by* OPT.

Definition 3. *The* **competitive ratio** *of an online algorithm for the* PROPHET *inequality problem or any of its generalizations is the maximum ratio between the prophet's expected reward and the online algorithm's expected reward*

$$\frac{\mathbb{E}[\mathsf{OPT}]}{\mathbb{E}[R]}.$$

Clearly, the online algorithm cannot do any better than the prophet, so the competitive ratio is at least 1, and a lower competitive ratio indicates a better online algorithm.

Note that some authors call the inverse of this the "competitive ratio". Fortunately, there is usually no ambiguity which definition is being used, because the inverse is at most 1.

3 The Original Prophet Inequality

Krengel and Sucheston [9] were the first to state the prophet inequality and show that the best possible competitive ratio is 2. The example that shows that the competitive ratio of 2 is best possible ([9], [8]) is as follows.

Example 1. Let there be two random variables X_1 and X_2. Let X_1 be 1 with probability 1, and let X_2 be n with probability $1/n$ and 0 otherwise.

- When an online algorithm observes X_1 to see w_1, it must decide whether to accept or reject it (or possibly make a randomized choice between the two), but since w_1 is fixed as always 1, the online algorithm has no new information about the actual values of the X_i to base this choice on. Therefore, the only two sensible online algorithms are "always pick X_1" or "always pick X_2". Clearly, both have expected value 1.
- On the other hand the prophet would just pick the largest value, which is n with probability $1/n$ and 1 with probability $(n-1)/n$, thus achieving an expected value of $2 - 1/n$.

Thus, letting $n \to \infty$ shows that the competitive ratio for the basic prophet inequality can not be better than 2.

Now, to prove the Prophet Inequality, we only need to exhibit online algorithms that achieve a competitive ratio of 2. Although Krengel and Sucheston were first to prove the basic Prophet Inequality, we will present two simpler proofs by later authors with explicit algorithms that achieve a competitive ratio of 2, which also demonstrate useful ideas for studying our generalizations. To introduce the explicit algorithms and lay the groundwork for understanding generalizations, we will discuss the important and general concept of "thresholds".

3.1 On Thresholds

In general, it's easy to see that any sensible online algorithm for the Prophet Inequality problem is determined by a *threshold function*, which maps each possible observed prefix $w_1, w_2, \ldots, w_i$ of the sequence of values to a *threshold* $T_{i+1} \in \mathbb{R}^+ \cup \{\infty\}$, such that X_{i+1} will be accepted iff $w_{i+1} \geq T$ (or iff $w_{i+1} > T$ — the two strategies are the same if the distribution of X_{i+1} has no point mass, but if not, there are technical differences that must be worked around.) This just means that given some prior observerations, if the online algorithm would have accepted some value, it would have accepted any higher value in the same position, which means the set of values it would have accepted in that position can be described by a single threshold. This is a natural consequence of the key propety that all the variables are independent: observing a higher value is always better, and will not negatively correlate with later variables. In all the generalizations we study, this observation will hold.

3.2 Constant Threshold Algorithms

One particularly simple class of algorithms for the original prophet inequality problem is those with *constant threshold*, for which the threshold function of every X_i is the same and does not depend on either the position or prior observations. In other words, the algorithm simply starts with a real number T computed from the distributions X_i and accepts the first value w_i that is $\geq T$ (or $> T$).

In fact, a constant threshold strategy is enough to achieve a competitive ratio of 2. Samuel-Cahn [13] was the first to propose and analyze a constant-threshold strategy; however, as part of their exposition, Kleinberg and Weinberg [8] present a different threshold that works just as well. Here we present both proofs in a unified fashion to emphasize their similarities.

Theorem 1. *As before, let* OPT *denote* $\max_i X_i$ *(i.e. the result obtained by the "prophet" or optimal offline algorithm) and denote the algorithm's reward by* R *(a random variable). Then:*

- *Let* m *be the median of the distribution of* OPT*, i.e. the value* m *such that*

  $$P[\mathsf{OPT} < m] \leq 1/2 \text{ and } P[\mathsf{OPT} > m] \leq 1/2.$$

  Then at least one of the algorithms "accept the first element with value $\geq m$*" and "accept the first element with value* $> m$*" achieves an expected value of* $\mathbb{E}[R] \geq \mathbb{E}[\mathsf{OPT}]/2$*, and thus competitive ratio 2.*

- *Let* t *be* $\mathbb{E}[\mathsf{OPT}]/2$*. Then the algorithm "accept the first element* $\geq t$*" achieves an expected value of* $\mathbb{E}[R] \geq \mathbb{E}[\mathsf{OPT}]/2$*, and thus competitive ratio 2.*

Intuitively, the proof breaks down the expected value of R into two parts

$$R = \min(R, T) + \max(R - T, 0):$$

the part below T (which is T if the algorithm finds a value above its threshold, and 0 if not) and the part above T (which is potentially unbounded depending on how lucky the algorithm is).[2][3] If OPT is high, then the part of R below T is guaranteed to be high because

[2] The second part $\max(R - T, 0)$ is sometimes denoted $(R - T)^+$, using the "positive clamp" function $(x)^+ := \max(x, 0)$. We won't use this notation here, but some of the referenced papers use it without definition.

[3] Many papers explore the connection between the prophet inequality and auctions. One way to frame

the algorithm will accept OPT or something before it, and the part of R above T is likely to be high because the algorithm has a fixed chance of not encountering any elements exceeding OPT before T.

We provide a formal proof below.

Proof. Let the random weights of the elements be $X_1, \ldots, X_n$, and consider the algorithm "accept the first of $X_1, \ldots, X_n$ that is observed to be $\underset{?}{>} T$", where "$\underset{?}{>}$" represents either "$>$" or "$\geq$", and T is a constant threshold, both to be chosen later.

R is the first element $X_c \underset{?}{>} T$ above the threshold, if one exists. If no such element exists, we assume the algorithm gets reward $R = 0$ for ease of analysis. (Since X_i are nonnegative, the algorithm in practice should always choose the last value and will do better.) Let p be the probability that the algorithm successfully accepts a value $R \underset{?}{>} T$.

As promised, we will look at the decomposition of $\mathbb{E}[R]$ into the part below T and the part above T:

$$\begin{aligned}\mathbb{E}[R] &= \int_0^\infty P[R > x]\,dx \\ &= \underbrace{\int_0^T P[R > x]\,dx}_{\mathbb{E}[\min(R,T)]} + \underbrace{\int_T^\infty P[R > x]\,dx}_{\mathbb{E}[\max(R-T,0)]}\end{aligned}$$

We bound these separately:

- **First part $\mathbb{E}[\min(R, T)]$:** We have

$$\mathbb{E}[\min(R, T)] = \int_0^T P[R > x]\,dx \geq pT$$

  ("the part of R below T") because when $x < T$, clearly $P[R > x] \geq P[R \underset{?}{>} T] = p$.

- **Second part $\mathbb{E}[\max(R - T, 0)]$:** We also have

$$P[R > x] \geq (1 - p) \sum_i P[X_i > x]$$

  because there is at least $(1 - p)$ probability that all X_j before X_i do not pass the threshold, and thus $(1 - p)P[X_i > x]$ probability that X_i will be the first value that surpasses the threshold, in which case the algorithm will choose X_i and will get reward $R = X_i > x$. Next, we have

$$\sum P[X_i > x] \geq P[\mathsf{OPT} > x]$$

the prophet inequality in terms of auctions is that you have one object that you are trying to sell to one of n bidders and you are trying to maximize your revenue. In this model, analyzing the threshold strategy involves privately declaring that the cost for producing the item was T, and separately analyzing the part of the revenue for recouping expenses plus the profit over the base cost.

by union-bound. Therefore,

$$\begin{aligned}\mathbb{E}[\max(R-T,0)] &= \int_T^\infty P[R>x]\,dx \\ &\geq (1-p)\int_T^\infty \sum_i P[X_i>x]\,dx \\ &\geq (1-p)\int_T^\infty P[\mathsf{OPT}>x]\,dx \\ &= (1-p)\mathbb{E}[\max(\mathsf{OPT}-T,0)].\end{aligned}$$

Putting the two bounds together to bound the value $\mathbb{E}[R]$, we have

$$\begin{aligned}\mathbb{E}[R] &= \underbrace{\int_0^T P[R>x]\,dx}_{\mathbb{E}[\min(R,T)]} + \underbrace{\int_T^\infty P[R>x]\,dx}_{\mathbb{E}[\max(R-T,0)]} \\ &\geq pT + (1-p)\underbrace{\int_T^\infty P[\mathsf{OPT}>x]\,dx}_{\mathbb{E}[\max(\mathsf{OPT}-T,0)]}.\end{aligned}$$

Now, separately observe that

$$\begin{aligned}T + \underbrace{\int_T^\infty P[\mathsf{OPT}>x]\,dx}_{\mathbb{E}[\max(\mathsf{OPT}-T,0)]} &\geq \underbrace{\int_0^T P[\mathsf{OPT}>x]\,dx}_{\mathbb{E}[\min(\mathsf{OPT},T)]} + \underbrace{\int_T^\infty P[\mathsf{OPT}>x]\,dx}_{\mathbb{E}[\max(\mathsf{OPT}-T,0)]} \\ &= \int_0^\infty P[\mathsf{OPT}>x]\,dx \\ &= \mathbb{E}[\mathsf{OPT}].\end{aligned}$$

For simplicity, let $\beta = \mathbb{E}[\max(\mathsf{OPT}-T,0)] = \int_T^\infty P[\mathsf{OPT}>x]\,dx$. Then, to summarize, we have shown above that

$$\mathbb{E}[R] \geq pT + (1-p)\beta \text{ and } T+\beta \geq \mathbb{E}[\mathsf{OPT}].$$

The proofs for the two different thresholds are now a few lines:

- (Samuel-Cahn [13]) Choose T to be a median of the distribution of OPT, so that $P[\mathsf{OPT}<T] \leq 1/2$ and $P[\mathsf{OPT}>T] \leq 1/2$. Then we can choose $\underset{?}{>}$ to be $>$ or $\geq$ and thus get $p \leq 1/2$ or $p \geq 1/2$, respectively.
  - If $T \leq \beta$, we choose $\underset{?}{>}$ to be $>$, so $p \leq 1/2$ and $(1-p) \geq 1/2$.
  - If $T \geq \beta$, we choose $\underset{?}{>}$ to be $\geq$, so $p \geq 1/2$ and $(1-p) \leq 1/2$.

  In both cases it immediately follows that

  $$\mathbb{E}[R] \geq pT + (1-p)\beta \geq (1/2)T + (1/2)\beta \geq \mathbb{E}[\mathsf{OPT}]/2,$$

  as desired.

- (Kleinberg and Weinberg [8]) Let $T = \mathbb{E}[\mathsf{OPT}]/2$. Then $\beta \geq \mathbb{E}[\mathsf{OPT}] - T = T$, so

$$\mathbb{E}[R] \geq pT + (1-p)\beta \geq pT + (1-p)T = T = \mathbb{E}[\mathsf{OPT}]/2$$

as desired.

$\square$

Combining this with the Example, we have now proven the basic Prophet Inequality, which is to say:

Theorem 2. *The optimal competitive ratio for the* Prophet *Inequality Problem is 2.*

Although all the results about generalizations are considerably more difficult and we will generally only sketch analyses of algorithms for them, many analyze threshold strategies with decompositions similar to the above analysis.

4 Prophet Inequalities with Choice of k Values

In the simplest generalization of the prophet inequality in the direction of allowing multiple choices, we fix some positive integer k and allow the prophet and algorithm to accept any k values instead of only one; we then define their reward to be the sum of the k accepted values and compare the ratio of the rewards. As mentioned in the introduction, when prophet inequalities are applied to sale mechanisms, this generalization arises naturally as the analogue of having multiple identical items for sale.

Definition 4. *Let k be a positive integer. An online algorithm for the* **k-Choice Prophet** *problem knows the distributions of $X_1, X_2, \ldots, X_n$ beforehand, but receives the actual values w_i of X_i online one at a time. Right after the algorithm receives the value w_i of some X_i, it must decide irrevocably whether to accept or reject X_i. The algorithm can accept at most k variables. After it has observed and made a decision on the last variable, the problem ends and the algorithm receives reward R equal to the sum of the values it has accepted.*

The reward of the prophet, still denoted by OPT*, is the sum of the largest k values w_i of distinct variables X_i.*

(Note that after the algorithm has accepted k variables, it must reject all the remaining variables no matter what it observes, but for simplicity and consistency with later definitions, we will still have the algorithm finish observing all of the variables.)

Before we continue, we note that even with the idea of allowing k values to be chosen, these are not the only choices of constraints and reward function, and they were not the first choices to be studied. Kennedy [10] studies the ratio between the best reward of a prophet that can accept k values and the best reward of an algorithm that can accept k' values, which is more general than our setup (where $k = k'$), but only gives significant results for the case $k = 1$. Assaf and Samuel-Cahn [2] also allow the online algorithm to accept k values, but consider the *maximum* of the k values instead of the sum as the reward, and analyzes its ratio to the prophet's best single choice (clearly, giving the prophet multiple choices in this setting does not help).

Hajiaghayi, Kleinberg, and Sandholm [6] were the first to obtain concrete bounds on the ratio between the best sum-of-value rewards of a prophet and an online algorithm, each of which can accept k values. They proved (Theorem 7) that the competitive ratio β_k is bounded by

$$1 + \sqrt{\frac{1}{512k}} \leq \beta_k \leq 1 + \sqrt{\frac{8 \ln k}{k}}.$$

The strategy they use to achieve the upper bound is also a simple threshold rule that accepts the first k elements it sees with value above T, where T is the infimum of thresholds satisfying

$$\sum P[x_i > T] \leq k - \sqrt{2k \ln k}.$$

The lower bound on the competitive ratio is obtained by considering $\sqrt{k/8}$ variables X_i that are always 1, followed by $2k$ variables that take values independently and uniformly at random from the set $\{0,2\}$.

The asymptotic gap in the competitive ratio was bridged by Alaei [1], who analyzes the same problem and obtains an improved competitive ratio of $1 + O(1/\sqrt{k})$, showing that this ratio is asymptotically tight. The proof is difficult to express in isolation because the paper reformulates it as another problem with a magician and wands that is useful for studying combinatorial auctions, but it also involves a threshold-like value T such that $\sum_i P[V_i > T] = k$.

The most recent result on this problem is due to Azar, Kleinberg, and Weinberg [3], who develop strategies for prophet inequalities that only need limited information about the distributions of the variables X_i. Their algorithm only requires one random sample drawn from the distributions of each of the variables (which are of course independent of the actual samples that it will encounter during the problem.) Their strategy is as follows.

"Rehearsal Algorithm": Randomly sample the distributions $\{X_i\}$. Given the samples sorted in decreasing order $s^{(1)} \geq \cdots \geq s^{(n)}$, make a list of k thresholds $T_1, T_2, \ldots, T_k$ where the top $k - 2\sqrt{k}$ thresholds $T_1, T_2, \ldots, T_{k-2\sqrt{k}}$ are equal to the top $k - 2\sqrt{k}$ samples $s^{(1)}, \ldots, s^{(k-2\sqrt{k})}$ and the bottom $2\sqrt{k}$ thresholds are all equal to the $(k - 2\sqrt{k})$th sample from the top $s^{(k-2\sqrt{k})}$. Now observe the variables in sequence. Maintain a list of "unmatched" thresholds; whenever a value is observed that is greater than an unmatched threshold, accept it and remove the largest unmatched threshold that is less than that value from the list.

Analysis of the algorithm is also rather complicated, reducing it to analyzing a random walk; we omit details.

5 Prophet Inequalities with Matroid Constraints

Further generalizations of multiple-choice prophet inequalities are possible. These generalizations are characterized by specifying a family of allowed "feasible sets", subsets of the $\{X_i\}$ that the algorithm is allowed to accept.

In the previous section, the family of feasible sets was just the family of sets with at most k elements. A common constraint that is more general than this is to take the family of feasible subsets to be a *matroid*, a combinatorial structure of subsets that mimics and generalizes linear independence of vectors in a vector space, among other structures seen in combinatorics.

Definition 5. *A* **matroid** *is a pair* $(\mathcal{U}, \mathcal{I})$ *where* $\mathcal{U}$ *is a finite set called the* ***ground set*** *and* $\mathcal{I}$ *is a family of subsets of* $\mathcal{U}$ *called the family of* ***independent subsets****, satisfying these properties:*

- *The empty set is independent.*
- *(**Hereditary property** or **Downward-closedness**) Any subset of an independent set is independent.*
- *(**Augmentation**) If A and B are two independent sets and A has fewer elements than B, then some element b of B can be added to A so that $A \cup \{b\}$ is still independent.*

The **rank** *of a matroid is the size of any maximal independent subset (it is the same for any maximal independent subset, which can quickly be seen from the augmentation property).*

While the definition of a matroid may seem somewhat artificial, they are commonly studied in optimization problems, which largely derives from the well-known "greediness property":

Proposition 1. *Consider a weight function on a matroid $w : \mathcal{U} \to \mathbb{R}^+$, which assigns a nonnegative weight to each element of the matroid's ground set, and the algorithmic task of finding the independent subset with the largest total weight. Such a subset can be found by the greedy algorithm, which at each step greedily takes the element of the ground set with greatest value that forms an independent set with the elements it has taken in previous steps.*

A well-known example of this greedy algorithm is Kruskal's algorithm for computing a minimum spanning tree. This is because in any graph, we can define a matroid with the graph's set of edges as its ground set, where a set of edges is independent iff it forms no cycles (i.e. is a forest). Such a matroid is called a *graphic matroid*.

Proof. Let the matroid's rank be k and let an independent subset with maximal weight be M. At the ith step, after the greedy algorithm has chosen i elements A, note that by applying the augmentation property $n-i$ times we could add $n-i$ elements from M to A while maintaining its independence. Thus, the element u of maximal weight that could be added to A while preserving its independence has weight at least that of the $(n-i)$th-smallest weight in M. Summing across i gives the desired result. □

In the *matroid prophet inequality problem*, the family of feasible subsets are the independent subsets of some matroid with the set of variables $\{X_i\}$ as its ground set.[4]

Definition 6. *An instance of the* **Matroid Prophet** *problem consists of a sequence of independent nonnegative random variables* $X_1, X_2, \ldots, X_n$ *as usual, plus a matroid* $\mathcal{I}$ *on the ground set* $\mathcal{U} = \{X_1, \ldots, X_n\}$.

An online algorithm for the Matroid Prophet *problem knows the distributions of* $X_1, X_2, \ldots, X_n$ *and the matroid* $\mathcal{I}$ *beforehand, but receives the actual values* w_i *of* X_i *online one at a time. Right after the algorithm receives the value* w_i *of some* X_i*, it must decide irrevocably whether to accept or reject* X_i*. At all times, the set of variables the algorithm has accepted must be an independent set in the matroid* $\mathcal{I}$*; it can only accept variables that maintain this property. After it has observed and made a decision on the last variable, the problem ends and the algorithm receives reward* R *equal to the sum of the values it has accepted.*

The reward of the prophet, still denoted by OPT*, is the maximum sum of values of variables in some indepndent set of the matroid* $\mathcal{I}$.

It is easy to verify that for any nonnegative integer k, one can obtain a matroid on any ground set by letting sets be independent iff their size is $\leq k$. This matroid is called the *uniform matroid of rank k*. Thus, Matroid Prophet is a generalization of k-Choice Prophet. Kleinberg and Weinberg [8] study the Matroid Prophet problem and shows the same tight competitive ratio 2 of the original Prophet problem is attainable. We begin discussing this proof with some definitions from their paper.

Definition 7. *Let* A *be a feasible subset of the variables* $\{X_i\}$*. Consider a random sequence of values* $w'_1, \ldots, w'_n$ *drawn from the distributions* X_i *and let the prophet's set of accepted variables for that instance be* B*. Then let the* remainder $\mathsf{Rem}(A)$ *of* A *be the*

[4] An analogous generalization of the secretary problem, the *matroid secretary problem*, can also be defined. It is correspondingly more well-studied than the matroid prophet inequality.

subset of B with maximum total value such that $A \cup \mathsf{Rem}(A)$ is still feasible, and let the cost $\mathsf{Cost}(A)$ *of A be the remaining elements of B,*

$$\mathsf{Cost}(A) = B \setminus \mathsf{Rem}(A).$$

Let $W'(\mathsf{Rem}(A))$ and $W'(\mathsf{Cost}(A))$ denote these subsets' total values computed from the aforementioned random sequence of values.

Note that both $\mathsf{Rem}(A)$ and $\mathsf{Cost}(A)$ are random variables, and if we let OPT' be the prophet's reward on the samples $\{w'_1, \ldots, w'_n\}$, then $\mathsf{Rem}(A) + \mathsf{Cost}(A) = \mathsf{OPT}'$. Also, in practice, A will be the set of accepted variables partway through the execution of our algorithm; despite this, the random sequence $\{w'_i\}$ in the definition above are entirely separate samples drawn from the problem instance's distributions, independent of whatever values w_i may have been observed by the algorithm.

Properties of matroids imply the following two lemmas, which we only have room to sketch proofs of. Still, note the usefulness of the greedy algorithm and property in reasoning about matroids.

Lemma 1. *For a fixed sequence of values $w'_1, \ldots, w'_n$ and definitions from Definition 7, the feasible $A \cup \mathsf{Rem}(A)$ is the feasible set with largest total value containing A. Thus, $\mathsf{Rem}(A)$ can be found by greedily adding elements of maximal weight to A that maintain its feasibility.*

Proof. (Sketch) If you run two greedy algorithms in parallel, one of which is committed to accepting A, then by matroid properties, the only time they make different decisions is when the first algorithm's choice would conflict with independence with A and is thus unavailable to the second algorithm. So the elements that would be greedily added to A are a subsequence of the elements that would be greedily added to the empty set. □

Lemma 2. $W'(\mathsf{Rem}(\cdot))$ *is a* submodular set function, *i.e.*

$$W'(\mathsf{Rem}(S)) + W'(\mathsf{Rem}(S \cup \{x,y\})) \le W'(\mathsf{Rem}(S \cup \{x\})) + W'(\mathsf{Rem}(S \cup \{y\}))$$

where all unions are disjoint set unions.

Intuitively, this states that if you force the prophet to accept two particular variables in conjunction, the combined forced loss is greater than the sum of the forced losses from needing to accept the two variables separately.

Proof. (Sketch) If you run two greedy algorithms committed to accepting S and $S \cup \{x\}$ in parallel, then by matroid properties, the former algorithm will make at most one acceptance of some w'_i different from the latter, which replaces x. If you also run two greedy algorithms committed to accepting $S \cup \{y\}$ and $S \cup \{x, y\}$ in parallel, the former algorithm will likewise make at most one acceptance of some w'_j different from the latter. Because of the way the restrictions interact, i is no earlier than j, so w'_i is the same or worse than w'_j; since both of these would be replacing x, the conclusion follows. □

The following property of an algorithm is a midpoint in Kleinberg and Weinberg's demonstration of an algorithm for the matroid secretary problem with competitive ratio 2:

Definition 8. *An online algorithm for the matroid prophet inequality problem has* 2-balanced thresholds *if, for every instance, if the algorithm ends with accepting variables A, and B is a set disjoint from A such that $A \cup B$ is feasible, then*

$$\sum_{x_i \in A} T_i \geq \frac{1}{2}\mathbb{E}[W'(\mathsf{Cost}(A))]$$
$$\sum_{x_i \in B} T_i \leq \frac{1}{2}\mathbb{E}[W'(\mathsf{Rem}(A))],$$

where T_i is the algorithm's threshold when it observed X_i and the expectations are over a sequence of values $\{w'_i\}$ drawn independently from $\{X_i\}$ described in Definition 7.

Here we see threshold functions being applied in analysis. Note that sometimes the matroid constraints imply the algorithm cannot accept a variable X_i no matter what its value w_i is; in those cases we set $T_i = \infty$.

Theorem 3. *If an online algorithm for the matroid prophet inequality problem has 2-balanced thresholds, then it achieves competitive ratio 2.*

Proof. (Sketch) The proof is similar to their proof for the original prophet inequality; for each accepted X_i, it decomposes the accepted value w_i into T_i and $w_i - T_i$; in other words,

$$\mathbb{E}[R] = \mathbb{E}\left[\sum_{X_i \in A} w_i\right] = \mathbb{E}\left[\sum_{X_i \in A} T_i\right] + \mathbb{E}\left[\sum_{X_i \in A} (w_i - T_i)\right].$$

The first term is bounded by the expected value of $\mathsf{Cost}(A)$ in the definition of 2-balanced thresholds. The second term is actually equal to

$$\mathbb{E}\left[\sum_{i=1}^{n} \max(w_i - T_i, 0)\right]$$

by the definition of thresholds, and can be bounded against $\mathsf{Rem}(A)$ using the second condition of 2-balanced thresholds. $\square$

After defining this property and proving that it is sufficient, Kleinberg and Weinberg demonstrate an algorithm exhibiting the property.

Theorem 4. *There exists an online algorithm for the matroid prophet inequality problem that has 2-balanced thresholds.*

A possibly more intuitive understanding of the algorithm we are about to describe is to imagine this two-stage extension to the matroid prophet inequality problem. The first stage is the same: the online algorithm observes values $w_1, \ldots, w_n$ and accepts a feasible set as usual, receiving reward equal to the sum of the accepted values. In the second stage, the algorithm observes an entirely new set of independent samples $w'_1, w'_2, \ldots, w'_n$ from the same distributions. Now, however, it gets to play the role of the prophet and choose which values to accept with knowledge of all values, but (1) it is committed to accepting the variables with their values from the first stage, and thus cannot select the same variables or any variables that fail to form a feasible set with them in the second stage; and (2) it gets only 1/2 of each value it accepts in the second stage. Then this online algorithm is one that accepts elements that improve its expected reward in this two-stage game. Of course, the second stage never actually occurs.

Proof. (Sketch) The algorithm accepts items into A such that the value

$$W(A) + \frac{1}{2}\mathbb{E}[W'(\mathsf{Rem}(A))]$$

is nondecreasing. Note that initially it is equal to

$$\frac{1}{2}\mathbb{E}[W'(\mathsf{Rem}(\emptyset))] = \frac{\mathbb{E}[\mathsf{OPT}]}{2},$$

which was the threshold in Kleinberg and Weinberg's proof of the original PROPHET inequality.

The first property of 2-balanced thresholds follows easily. The second property involves invoking several properties of matroids to show that we can construct an injective map from any B into $\mathsf{Rem}(A)$ such that every element of B could be replaced with the corresponding element of $\mathsf{Rem}(A)$ while only increasing its total value (i.e. as per Lemma 1, $\mathsf{Rem}(A)$ is an "optimal" set to add to A and greedily using parts of it can give partial optimality), and then applying Lemma 2 that $W'(\mathsf{Rem}(\cdot))$ is a submodular set function to compare the relevant sums. $\square$

A further generalization of matroid restrictions worth mentioning is to take the family of feasible subsets to be the intersection of multple matroids (a "polymatroid"), which is studied by Kleinberg and Weinberg [8] as above, as well as by Dütting and Kleinberg [5]. The analysis involves more general α-balanced thresholds. We omit details.

6 Prophet Inequalities with General Constraints

The most general possible multiple-choice constraint one could define on a prophet inequality problem places no constraints on the family of feasible sets of all, except that they are "downward-closed" (so if you're allowed to accept a certain set of variables, you could have accepted any subset of those variables.)[5]

Definition 9. *A* **downward-closed family of subsets** $\mathcal{F}$ *of a set* $\mathcal{U}$ *is a family of sets such that:*

- *The empty set is in* $\mathcal{F}$.
- *Any subset of a set in* $\mathcal{F}$ *is in* $\mathcal{F}$.

In analogy with matroids, we will say that the **rank** *of a downward-closed family is the maximum size of a set in the family.*

Definition 10. *An instance of the* **Downward-Closed Prophet** *problem consists of a sequence of independent nonnegative random variables* $X_1, X_2, \ldots, X_n$ *as usual, plus a downward-closed family of subsets* $\mathcal{F}$ *on* $\{X_1, \ldots, X_n\}$*, elements of which are called* feasible sets*.*

An online algorithm for the Downward-Closed Prophet *problem knows the distributions of* $X_1, X_2, \ldots, X_n$ *and the downward-closed family* $\mathcal{F}$ *beforehand, but receives the actual values* w_i *of* X_i *online one at a time. Right after the algorithm receives the value* w_i *of some* X_i*, it must decide irrevocably whether to accept or reject* X_i*. At all times, the set of variables the algorithm has accepted must be a feasible set in* $\mathcal{F}$*; it can only accept variables that maintain this property. After it has observed and made a decision on the last variable, the problem ends and the algorithm receives reward* R *equal to the sum of the values it has accepted.*

The reward of the prophet, still denoted by OPT*, is the maximum sum of values of variables in some feasible set.*

Rubenstein [12] constructs an algorithm for Downward-Closed Prophet that demonstrates a competitive ratio of $O(\log n)$ on n variables drawn from $\{0,1\}$, which can be converted into an algorithm that achieves a competitive ratio of $O(\log n \cdot \log k)$ on n variables

[5] Actually, Rubenstein [12] also briefly examines prophet inequalities for the problem Non-Monotone Prophet where the family of feasible sets does not even need to be downward-closed. In this generalization, the online algorithm is allowed to have accepted an infeasible set of variables in the middle of the problem, but must have accepted a feasible set at the end. This generalization is not very interesting — the structure of feasible sets can force the online algorithm to precommit to one of $\Theta(n)$ random variables without any information about them, and the prophet easily gets $\Theta(n)$ advantage in that case, n being the number of variables.

drawn from arbitrary nonnegative distributions where the largest feasible subset has size k. We will first explain this conversion:

Theorem 5. *Given an algorithm for* Downward-Closed Prophet *with random variables taking values in* $\{0,1\}$ *that achieves competitive ratio* $O(\log n)$*, we can obtain an algorithm for* Downward-Closed Prophet *with random variables of arbitrary distributions that achieves competitive ratio* $O(\log n \cdot \log k)$*, where* k *is the size of the largest feasible set.*

Proof. We will classify all values (not variables!) the prophet accepts into buckets. First note that we can ignore all values less than $\mathsf{OPT}/2r$ while only losing a constant factor in our competitive ratio, since the total reward in any scenario obtained from accepting such values can never exceed $\mathsf{OPT}/2$. Now, we classify values into the $\log k + 3$ intervals

$$[\mathsf{OPT}/2k, \mathsf{OPT}/k], [\mathsf{OPT}/k, 2\mathsf{OPT}/k], \ldots, [\mathsf{OPT}, 2\mathsf{OPT}], [2\mathsf{OPT}, \infty),$$

and look at the bucket with the largest expected contribution to the prophet's reward, which is at least $\Omega(1/\log k)$ fraction of it. To complete the proof, we only have to show we can obtain an $\Omega(1/\log n)$ fraction of the expected prophet reward obtained from values in this bucket, possibly using the $O(\log n)$-competitive algorithm. There are two cases:

- ("Tail") Values in the interval $[2\mathsf{OPT}, \infty)$ have the largest expected contribution. In this case, we will just accept all values that are at least $2\mathsf{OPT}$. There is at least $1/2$ chance that no values above $2\mathsf{OPT}$ will appear, so we can derive a constant bound on the probability that we see a value above $2\mathsf{OPT}$ but can't take it due to feasibility constraints. Thus we will obtain a *constant* fraction of the expected contribution of values in this bucket.

- ("Core") Values in some finite interval $[\ell, 2\ell]$ have the largest expected contribution. In this case, we will treat values in that interval as 1 and all other values as 0 and run the $O(\log n)$-competitive algorithm. When this algorithm obtains reward R, we obtain reward at least ℓR, which loses at most a constant factor when comparing competitive ratios.

□

We now turn to the strategy for random variables in $\{0,1\}$. First we note we can assume $\mathbb{E}[\mathsf{OPT}] > \log n$:

Proposition 2. *If* $\mathbb{E}[\mathsf{OPT}] = O(\log n)$*, the trivial greedy algorithm achieves a competitive ratio of* $O(\log n)$*.*

This is because the trivial greedy algorithm gets at least 1 reward in any sequence of observations that is not all 0s, which is enough.

The analysis for the general case depends on the idea of a "restricted prophet". A *restricted prophet* is a prophet that has committed to either accept or reject each of the values up to some point in the sequence, but can choose which of the remaining values to accept with full knowledge of all values. While our online algorithm executes, it will consider the restricted prophet that has committed to make the same acceptances or rejections that the algorithm already has. Of course, the online algorithm does not know the restricted prophet's actual reward, but can compute its expected reward over the values that the online algorithm hasn't seen.

Note the similarity to the remainder and cost analyzed in the previous section, but also note the difference: these restricted prophets are committed to accepting and rejecting variables as the algorithm has decided, while in the previous section they are only commmitted to accepting variables the algorithm has accepted, and could accept or reject variables the algorithm has rejected.

Throughout its execution, the algorithm maintains a target value τ and a target probability π, where π is the probability that the restricted prophet will achieve reward above τ. We need an analyzable proxy for which variables are good to take:

Definition 11. *Let τ and π be the values described above at some point in an online algorithm's execution. A future variable X_i is* **good** *if, under the condition it is the next 1 observed, the restricted prophet that accepts it (in addition to being committed to all decisions made by the online algorithm so far) has at least π/n^2 probability of achieving award at least τ. Otherwise, it is* **bad**.

Basically, the set of good variables are the variables whose chances of being 1 contribute "significantly" to the restricted prophet's probability of beating τ. Note that whether a variable is good depends on the algorithm's current state, including the previously observed values, acceptances, rejections, and τ.

Let G be the random event that at least one good feasible variable will be observed to have value 1 in the future. Also as before, let A denote the set of accepted variables.

Note that since we're only dealing with values in $\{0,1\}$ and there's no point in accepting 0 values, the number $|A|$ of accepted variables in any optimal algorithm will just be the collected reward so far. However, in Kleinberg and Weinberg's analysis, they have the algorithm possibly "hallucinate" that some 0s it observes are actually 1s in the corner case where no more good 1s are observed, for ease of analysis. In that case we still use $|A|$ as our accumulated reward in analysis. We maintain that, on expectation, only a constant fraction of the variables we have accepted are hallucinated 1's, so that we have a constant fraction of "real" 1's on expectation and we only lose a constant competitive ratio.

The factor of n^2 we are willing to tolerate in a good variable in Definition 11 seems like a lot, but it turns out that "most of the time", decreasing τ by 1 increases π by a factor of 2, so we only need to decrease τ by $\log n$ to compensate; and since we are allowed to assume $\mathsf{OPT} = \omega(\log n)$ by Proposition 2, this decrease is fine. We more formally describe the algorithm below and give a more detailed sketch of its analysis:

Algorithm. Let τ be a target reward initialized to $\mathbb{E}[\mathsf{OPT}]/2$. When $P[G] \geq 1/3$, observe values until you observe a good variable with value 1, and accept it. After each newly accepted element, decrease τ until $P[G] \geq 1/3$ or $|A| > \tau$ (in which case we are already guaranteed more reward than our target τ, modulo "hallucinations".)

If by the last good variable, no good 1s have been observed (so G has failed to occur), then "hallucinate" that that last good variable has value 1, accept it (so that $|A|$ is incremented), and decrease τ by 1.

We now show the lemma achieves an $O(\log n)$ competitive ratio in a series of lemmas. The first lemma shows that we start the algorithm in a good enough state:

Lemma 3. *The initial value of π is*

$$P\left[\mathsf{OPT} \geq \frac{\mathbb{E}[\mathsf{OPT}]}{2}\right] > 1/3 - o(1).$$

Proof. (Sketch) We decompose $\mathbb{E}[\mathsf{OPT}]$ similarly as before:

$$\begin{aligned}
\mathbb{E}[\mathsf{OPT}] &= \int_0^\infty P\left[\mathsf{OPT} \geq t\right]\, dt \\
&= \int_0^{\mathbb{E}[\mathsf{OPT}]/2} P\left[\mathsf{OPT} \geq t\right]\, dt + \int_{\mathbb{E}[\mathsf{OPT}]/2}^{2\mathbb{E}[\mathsf{OPT}]} P\left[\mathsf{OPT} \geq t\right]\, dt + \int_{2\mathbb{E}[\mathsf{OPT}]}^{\infty} P\left[\mathsf{OPT} \geq t\right]\, dt.
\end{aligned}$$

The first two terms are easy to bound as under

$$\frac{\mathbb{E}[\mathsf{OPT}]}{2} \text{ and } \frac{3\mathbb{E}[\mathsf{OPT}]}{2} \cdot P\left[\mathsf{OPT} \geq \frac{\mathbb{E}[\mathsf{OPT}]}{2}\right],$$

just by taking the obvious upper bounds of their integrands. The third term is more difficult to bound, but based on an inequality by Ledoux [11], turns out to be $O\left(e^{-\mathsf{OPT}}\right) = o(1)$. Rearranging terms yields the desired bound. ∎

Now we show that lowering our target τ substantially improves our chances π of hitting the target:

Lemma 4. *If $P[G] \leq 1/3$, then subtracting 1 from τ at least doubles π.*

Proof. (Sketch) First, across choices of the unknown variables, the restricted prophet has probability π of achieving reward above τ; we claim that less than π/n (i.e. $o(1)$ fraction) of this probability comes from the case where G is false. This is true because if no good variables have value 1, the first 1 selected by the restricted prophet decreases π by a factor of n^2, so a union-bound across which variable is the first 1 selected by the prophet gives the desired result.

Next, we claim that, conditioned on G, the restricted prophet's expected reward increases by at most 1. To see this, note that G is a disjoint union of events of the form

$$G^j = \text{“}X_j \text{ is the next variable to be observed with value 1”}$$

for good variables X_j. For each G^j, the restricted prophet's expected reward conditioned on G_j is at most 1 more than its reward conditioned on

$$B^j = \text{“All variables up to and including } X_j \text{ are observed to have value 0”},$$

which is no better than its reward conditioned on nothing at all.

The lemma follows from these two claims, because the probability of achieving $\tau - 1$ is at least $(3 - o(1))$ times the probability of achieving τ. □

Now we can complete the analysis of the algorithm. At the start of the algorithm, π is at least 1/4 by Lemma 3. Whenever we accept a variable, since we only accept good variables, we know that π decreases by at worst a factor of $\Omega(1/n^2)$ (and, in particular, never drops to 0). Whenever we decrease τ, π doubles. We can obtain a desired bound on τ from $\pi \leq 1$ since it's a probability. This gives us a bound on the number of accepted variables $|A|$ at the end of the problem: because there are no good variables (or, indeed, any variables at all) and no uncertainty left, $\pi > 0$ implies $\pi = 1$, so it can only be that $|A| \geq \tau$. Sometimes we accept “hallucinated” 1's, but since we only wait for 1's when we have at least a 1/3 chance of getting one, we expect that at least 1/3 of our accepted variables are non-hallucinatory 1's, and thus our actual reward is $\Omega(1)|A|$ on expectation.

This implies the competitive ratio of $O(\log n)$ for Downward-Closed Prophet with $\{0,1\}$ variables, and thus $O(\log n \cdot \log k)$ for Downward-Closed Prophet with general variables.

7 Conclusion

We surveyed multiple-choice prophet inequalities. We saw the importance of thresholds to all algorithms for the class of problems, and the power of simple algorithms with simple thresholds, as well as the common technique in analysis of decomposing values into parts above and below a threshold. Although the classical prophet inequality and some of the generalizations we surveyed have been solved and their optimal competitive ratios determined a long time ago, many interesting gaps in bounds on competitive ratios remain in Downward-Closed Prophet and other generalizations we have only been able to touch on. These is only the tip of the iceberg with regards to generalizations, even multiple-choice ones; we have not gone into studies of more specific types of matroids or generalizations where the variables can be reordered by the algorithm or by a dynamic adversary, for example.

This table summarizes the competitive ratios of the problems we have surveyed:

Problem	Discussed in	Competitive Ratio Lower bound	 Upper bound
PROPHET	Section 3	2 [9]	2 [9]
k-CHOICE PROPHET	Section 4	$1 + \Omega(1/\sqrt{k})$ [6]	$1 + O(1/\sqrt{k})$ [1]
MATROID PROPHET	Section 5	2 [9]	2 [8]
DOWNWARD-CLOSED PROPHET	Section 6	$\Omega(\log n / \log \log n)$ [4]	$O(\log n \cdot \log k)$ [12]

References

[1] Alaei, Saeed. "Bayesian Combinatorial Auctions: Expanding Single Buyer Mechanisms to Many Buyers." *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 512–521. 2011.

[2] Assaf, David and Ester Samuel-Cahn. "Simple ratio prophet inequalities for a mortal with multiple choices." *J. Appl. Probab. Volume 37, Number 4* (2000), pages 1084–1091. 2000.

[3] Azar, Pablo D., Robert Kleinberg, and S. Matthew Weinberg. "Prophet Inequalities with Limited Information." 2013.

[4] Babaioff, Moshe, Nicole Immorlica, and Robert Kleinberg. "Matroids, secretary problems, and online mechanisms." *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, SODA 2007*, pages 434–443, 2007.

[5] Dütting, Paul, and Robert Kleinberg. "Polymatroid Prophet Inequalities." arXiv:1307.5299. 2013.

[6] Hajiaghayi, MohammadTaghi, Robert Kleinberg, and Tuomas Sandholm. "Automated Online Mechanism Design and Prophet Inequalities". In *AAAI*, pages 58–65, 2007.

[7] Hill, Theodore P., and Robert P. Kertz. "A Survey of Prophet Inequalities in Optimal Stopping Theory." Contemporary Mathematics Volume 125, 1992.

[8] Kleinberg, Robert, and S. Matthew Weinberg. "Matroid Prophet Inequalities." arXiv:1201.4764. 2012.

[9] Krengel, Ulrich, and Louis Sucheston. "On semiamarts, amarts, and processes with finite value." Adv. in Prob. Related Topics, 4:197–266, 1978.

[10] Kennedy, D. P. "Prophet-Type Inequalities for Multi-Choice Optimal Stopping." *Stochastic Processes and their Applications* 24 (1987) 77-88. 1987.

[11] Ledoux, Michal. "On Talagrand's deviation inequalities for product measures." *ESAIM: Probability and Statistics*, 1:63–87, 997.

[12] Rubenstein, Aviad. "Beyond matroids: secretary problem and prophet inequality with general constraints." STOC 2016: 324–332. 2016.

[13] Samuel-Cahn, Ester. "Comparison of Threshold Stop Rules and Maximum for Independent Nonnegative Random Variables." *The Annals of Probability* Vol. 12, No. 4 (Nov., 1984), pages 1213–1216. 1984.