Sequential Decision Making: Prophets and Secretaries I - Prophet Inequalities

Matt Weinberg

Princeton University

Online Selection Problems

Imagine you’re trying to *hire a secretary*, find a job, select a life partner, etc.

- **At each time step:**
  - A secretary* arrives.
    - *I’m really sorry, but for this talk the secretaries will be pokémon.
  - You interview, learn their value.
  - Immediately and irrevocably decide whether or not to hire.
  - May only hire one secretary!

t = 1 2 3

An Impossible Problem

Offline:

- Every secretary i has a weight w_i (chosen by adversary, unknown to you).
- Adversary chooses order to reveal secretaries.

Online:

- Secretaries revealed one at a time. You learn their weight.
- Immediately and irrevocably decide to hire or not.
- May only hire one secretary!

Goal: Maximize probability of selecting max-weight element.

Trivial lower bound: can't beat 1/n (hire random secretary).

w = 6 4 7 8 9

Online Selection Problems: Secretary Problems

Offline:

- Every secretary i has a weight w_i (chosen by adversary, unknown to you).
- Secretaries permuted randomly.

Online:

- Secretaries revealed one at a time. You learn their weight.
- Immediately and irrevocably decide to hire or not.
- May only hire one secretary!

Goal: Maximize probability of selecting max-weight element.

w = 6 4 7 8 9

Online Selection Problems: Secretary Problems

Offline:

- Every secretary i has a weight w_i (chosen by adversary, unknown to you).
- Secretaries permuted randomly.

Online:

- Secretaries revealed one at a time. You learn their weight.
- Immediately and irrevocably decide to hire or not.
- May only hire one secretary!

Goal: Maximize probability of selecting max-weight element.

t = 1 2 3

w = 7 6 8

Online Selection Problems: Prophet Inequalities

Offline:

- Every secretary i has a weight w_i drawn independently from distribution D_i.
- Adversary chooses distributions and ordering (both known to you).

Online:

- Secretaries revealed one at a time. You learn their weight.
- Immediately and irrevocably decide to hire or not.
- May only hire one secretary!

Goal: Maximize expected weight of selected element.

w = U[4,6] U[0,8] U[3,4] U[4,5] U[0,9]

Online Selection Problems: Prophet Inequalities

Offline:

- Every secretary i has a weight w_i drawn independently from distribution D_i.
- Adversary chooses distributions and ordering (both known to you).

Online:

- Secretaries revealed one at a time. You learn their weight.
- Immediately and irrevocably decide to hire or not.
- May only hire one secretary!

Goal: Maximize expected weight of selected element.

t = 1

w = 5 U[0,8] U[3,4] U[4,5] U[0,9]

Online Selection Problems: Prophet Inequalities

Offline:

- Every secretary i has a weight w_i drawn independently from distribution D_i.
- Adversary chooses distributions and ordering (both known to you).

Online:

- Secretaries revealed one at a time. You learn their weight.
- Immediately and irrevocably decide to hire or not.
- May only hire one secretary!

Goal: Maximize expected weight of selected element.

t = 1 2

w = 5 7 U[3,4] U[4,5] U[0,9]

Prophet Inequalities

Observation: can find optimal policy via dynamic programming/backwards induction.

- If we make it to Mewtwo, clearly we should accept.
- If we make it to Pikachu, we can either:

  Reject: Get 4.5 from Mewtwo.

  Accept: Get w(Pikachu).

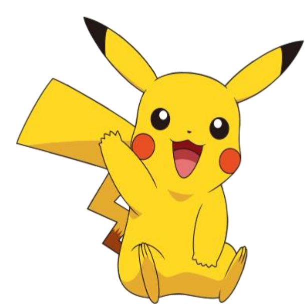
  So accept iff w(Pikachu) > 4.5.
- If we make it to Charmander, we can either:

  Reject: Get 4.625 (from optimal policy starting @ Pikachu).

  Accept: Get w(Charmander).

  So reject Charmander.
- Etc.




w = U[4,6] U[0,8] U[3,4] U[4,5] U[0,9]

Prophet Inequalities

Observation: can find optimal policy via dynamic programming/backwards induction.

- Question 1: How well does this policy do compared to a "prophet?"
  - Exist c such that for all instances, E[Gambler] $\geq$ c·E[Prophet]?
- Question 2: How well do "simpler" policies do?
  - Ex: set threshold T, accept first element with weight > T?
  - Can we get the same c as above?

Gambler
knows distributions,
uses online policy

vs.

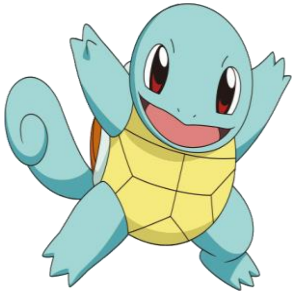
Prophet
knows weights,
picks best element

Prophet Inequalities

Theorem [Krengel-Sucheston 78, Samuel-Cahn 86]: Uniform threshold guarantees E[Gambler] $\geq 1/2 \cdot$E[Prophet]. Best possible (for all policies).

Tight example:

- Prophet gets $1/\epsilon$ w.p. ϵ, 1 w. p. 1-ϵ. E[prophet] = $2 - \epsilon$.
- Gambler can accept Bulbasaur, get 1.
- Or reject and get Squirtle, also for 1. So E[gambler] = 1.



w = 1

0, w.p. 1-ϵ
$1/\epsilon$, w.p. ϵ

Prophet Inequalities

Theorem [Krengel-Sucheston 78, Samuel-Cahn 86]: Uniform threshold guarantees E[Gambler] ≥ 1/2 ·E[Prophet]. Best possible (for all policies).

(modified) Proof:

- Let T $= E[\max_i\{w_i\}/2]$, use threshold T (accept any element > T).
- Define $p = \Pr[\max_i\{w_i\} > T]$. Define $ALG_i = w_i \cdot I(Alg\ accepts\ i)$.
- Notation: $X^+ = \max\{X, 0\}$.

$E[ALG] = \sum_i E[ALG_i] = \sum_i E[(T + w_i - T) \cdot I(Alg\ accepts\ i)]$.

$= pT + \sum_i E[(w_i - T) \cdot I(Alg\ accepts\ i)]$.

$= pT + \sum_i E[(w_i - T) \cdot I(w_i > T \text{ AND don't accept any j} < \text{i})]$.

$= pT + \sum_i E[(w_i - T)^+ \cdot I(\text{Don't accept any j} < \text{i})]$.

$= pT + \sum_i E[(w_i - T)^+] \cdot \Pr[\text{Don't accept any j} < \text{i}]$.

$\geq pT + (1 - p) \sum_i E[(w_i - T)^+]$.

So: A) $\boldsymbol{E[ALG] \geq pT + (1 - p) \sum_i E[(w_i - T)^+]}$.

Prophet Inequalities

Theorem [Krengel-Sucheston 78, Samuel-Cahn 86]: Uniform threshold guarantees E[Gambler] $\geq 1/2$ ·E[Prophet]. Best possible (for all policies).

(modified) Proof:

So: A) $\boldsymbol{E[ALG] \geq pT + (1-p)\sum_i E[(w_i - T)^+]}$.

Just need to bound $E\left[\max_i\{w_i\}\right]$. Recall $\mathrm{T} = E\left[\max_i\{w_i\}\right]/2$.

$$E\left[\max_i\{w_i\}\right] \leq E\left[T + \left(\max_i\{w_i\} - T\right)^+\right].$$

$$\leq T + E[\max_i\{(w_i - T)^+\}]\ .$$

$$\leq T + E[\textstyle\sum_i (w_i - T)^+].$$

$$\Rightarrow \textstyle\sum_i E[(w_i - T)^+] \geq E\left[\max_i\{w_i\}\right] - T = E[\max_i\{w_i\}]/2.$$

So: B) $\boldsymbol{T = E[max_i\{w_i\}]/2}$, $\boldsymbol{\sum_i E[(w_i - T)^+] \geq E[max_i\{w_i\}]/2}$.

$\Rightarrow \boldsymbol{E[ALG] \geq E[max_i\{w_i\}]/2}$.

Prophet Inequalities

Theorem [Krengel-Sucheston 78, Samuel-Cahn 86]: Uniform threshold guarantees E[Gambler] $\geq 1/2 \cdot$E[Prophet]. Best possible (for all policies).

(modified) Proof:

A) $\boldsymbol{E[ALG] \geq pT + (1-p)\sum_i E[(w_i - T)^+]}$.

B) $\boldsymbol{T = E[max_i\{w_i\}]/2}$, $\boldsymbol{\sum_i E[(w_i - T)^+] \geq E[max_i\{w_i\}]/2}$.

$\Rightarrow \boldsymbol{E[ALG] \geq E[max_i\{w_i\}]/2}$.

Intuition: A) holds for **any** T. B) lets us get mileage from A).

Because T **not too big**, $\sum_i E[(w_i - T)^+] \geq E\left[\max_i\{w_i\}\right]/2$.

Because T **not too small**, $\mathrm{T} \geq E\left[\max_i\{w_i\}\right]/2$.

T is a **balanced threshold** (not formal definition yet).

Multiple Choice Prophet Inequalities

We just saw:

- Simple description of optimal stopping rule.
- Tight competitive analysis, also achieved by uniform threshold.

Rest of talk: What if multiple choices?

Offline:

- Secretary i has a weight w_i drawn independently from distribution D_i.
- Adversary chooses distributions, ordering, and **feasibility constraints**: which secretaries can simultaneously hire? (all known to you)

Online:

- Secretaries revealed one at a time. You learn their weight.
- Immediately and irrevocably decide to hire or not.
- H = all hired secretaries. Must maintain H feasible **at all times**.

Goal: Maximize $\mathrm{E}[\sum_{i\in H} w_i]$ - expected weight of hires.

Multiple Choice Prophet Inequalities

Examples:

- Feasible to hire any k secretaries (k-uniform matroid).
- Associate each secretary with an edge in a graph. Feasible to hire any acyclic subgraph (graphic matroid).
- Associate each secretary with a vector in a vector space. Feasible to hire any linearly independent subset (representable matroid).
- Associate each secretary with an edge in a bipartite graph. Feasible to hire any matching (intersection of two partition matroids).

w = U[4,6] U[0,8] U[3,4] U[4,5] U[0,9]

State-of-the-art (non-exhaustive)

Feasibility	Approximation Guarantee
k-Uniform	Algorithm: $1+O(1/\sqrt{k})$ [**Alaei 11**]. Lower Bound: $1+\Omega(1/\sqrt{k})$ [**Kleinberg 05**].
Matroids	Algorithm: 2 [**Kleinberg-W. 12**]. Lower Bound: 2.
Intersection of P Matroids	Algorithm: 4P-2 [**KW 12**]. Lower Bound: P+1 [**KW 12**].
Arbitrary Downwards Closed	Algorithm: O(log n log r) [**Rubinstein 16**]. Lower Bound: Ω(log n/log log n) [**Babaioff-Immorlica-Kleinberg 07**]. n = #elements, r = size of largest feasible set.
Independent set in Graph	Algorithm: $O(\rho^2 \log n)$ [**Gobel-Hoefer-Kesselheim-Schleiden-Vocking 14**]. Lower Bound: $\Omega(\log n/\log^2(\log n))$ [**GHKSV 14**]. ρ = "inductive independence number" of graph.
Polymatroids	Algorithm: 2 [**Dutting-Kleinberg 15**]. Lower Bound: 2.

Matroid: S, T feasible, |S|>|T| → $\exists i \in S, T \cup \{i\}$ feasible. Downwards closed.

Think: feasible ≈ linearly independent in a vector space.

Matroid Intersection: ∃ P matroids $M_1, \ldots, M_P$, S feasible ↔ S feasible in each M_i.

Bipartite matchings = intersection 2 matroids. 3D matchings = 3 matroids.

Rest of Talk – Balanced Thresholds

Goal: Introduce concept of “balanced thresholds” via:

- Formal definition.
- 2-approximation for k-uniform [Chawla-Hartline-Malec-Sivan 10].
- 2-approximation for matroids (partial analysis).

Recall high level idea:

- Want thresholds **big enough** so that thresholds themselves contribute high weight.
- Want thresholds **small enough** so that expected surplus still high.

Balanced Thresholds – Cost and Remainder

Notation: OPT($w_1, \dots, w_n$) = max-weight feasible set.

- Will drop ($w_1, \dots, w_n$), just remember that OPT depends on weights.

Definition: Remainder(H,$w_1, \dots, w_n$) = $\underset{S \subseteq OPT, S \cup H \text{ feasible}}{\operatorname{argmax}} \{\sum_{i \in S} w_i\}$.

- "Best subset of OPT that could have added to H."

Definition: Cost(H,$w_1, \dots, w_n$) = OPT – Remainder(H).

- "What we lost from OPT by accepting H."

- Will abuse notation. Use OPT, Remainder, Cost to refer to these **sets**. As well as their **weights**.

Balanced Thresholds – Cost and Remainder

Definition: Remainder$(\mathrm{H}, w_1, \ldots, w_n) = \underset{S \subseteq OPT, S \cup H \text{ feasible}}{\operatorname{argmax}} \{\sum_{i \in S} w_i\}$.

- "Best subset of OPT that could have added to H."

Definition: Cost$(\mathrm{H}, w_1, \ldots, w_n)$ = OPT – Remainder(H).

- "What we lost from OPT by accepting H."

Example: Sets of size 1 feasible. OPT = {Mewtwo}.

Remainder({Charmander}) = Ø. Cost({Charmander}) = {Mewtwo}.

Remainder(Ø) = {Mewtwo}. Cost(Ø) = Ø.

w = 1 2 3 4 5

Balanced Thresholds – Cost and Remainder

Definition: Remainder(H,$w_1, \ldots, w_n$) = $\underset{S \subseteq OPT, S \cup H \text{ feasible}}{\operatorname{argmax}} \{\sum_{i \in S} w_i\}$.

- “Best subset of OPT that could have added to H.”

Definition: Cost(H,$w_1, \ldots, w_n$) = OPT – Remainder(H).

- “What we lost from OPT by accepting H.”

Example: Sets of size 2 feasible. OPT = {Mewtwo, Pikachu}.

Remainder({Charmander}) = {Mewtwo}. Cost({Charmander}) = {Pikachu}.

Remainder(∅) = {Mewtwo, Pikachu}. Cost(∅) = ∅.

Remainder({Bulbasaur, Squirtle}) = ∅. Cost({Bulbasaur, Squirtle}) = {Mewtwo, Pikachu}.

w = 1 2 3 4 5

Balanced Thresholds – Cost and Remainder

Definition: Remainder(H,$w_1, \ldots, w_n$) = $\underset{S \subseteq OPT, S \cup H \text{ feasible}}{\operatorname{argmax}} \{\sum_{i \in S} w_i\}$.

- "Best subset of OPT that could have added to H."

Definition: Cost(H,$w_1, \ldots, w_n$) = OPT – Remainder(H).

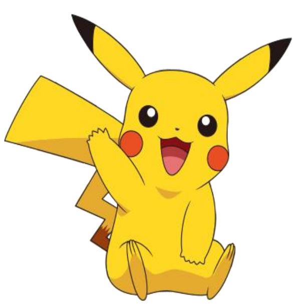
- "What we lost from OPT by accepting H."

Example: Sets of size k feasible. OPT = top k elements.

Remainder(H) = top k-|H| elements.

Cost(H) = lowest |H| elements of top k.




w = 1 2 3 4 5

Balanced Thresholds – Cost and Remainder

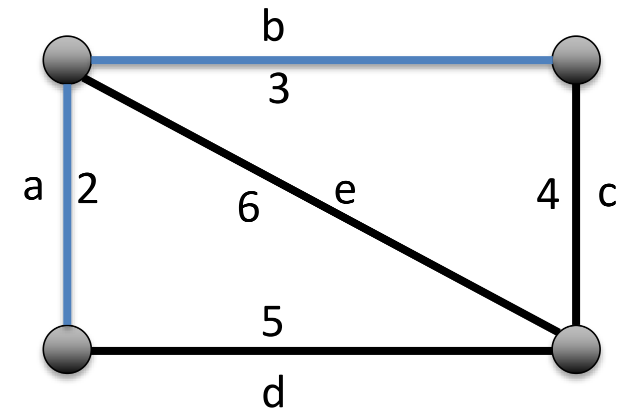
Definition: Remainder(H,$w_1, \ldots, w_n$) = $\underset{S \subseteq OPT, S \cup H \text{ feasible}}{\operatorname{argmax}} \{\sum_{i \in S} w_i\}$.

- "Best subset of OPT that could have added to H."

Definition: Cost(H,$w_1, \ldots, w_n$) = OPT – Remainder(H).

- "What we lost from OPT by accepting H."

OPT = {e, d, c}.



Balanced Thresholds – Cost and Remainder

Definition: Remainder(H,$w_1, \dots, w_n$) = $\underset{S \subseteq OPT, S \cup H \text{ feasible}}{\operatorname{argmax}} \{\sum_{i \in S} w_i\}$.

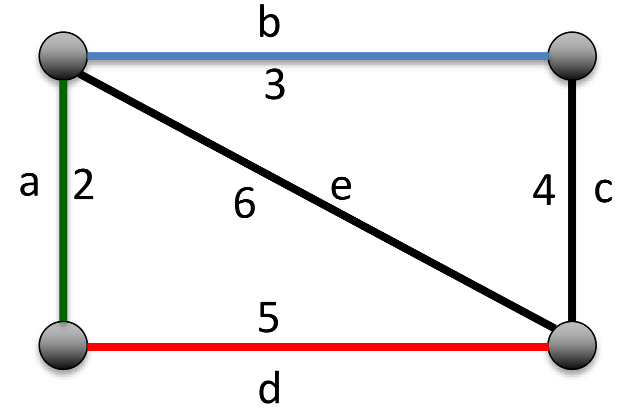
- "Best subset of OPT that could have added to H."

Definition: Cost(H,$w_1, \dots, w_n$) = OPT – Remainder(H).

- "What we lost from OPT by accepting H."

OPT = {e, d, c}.

Remainder({a}) = {e,c}. Cost({a}) = d.



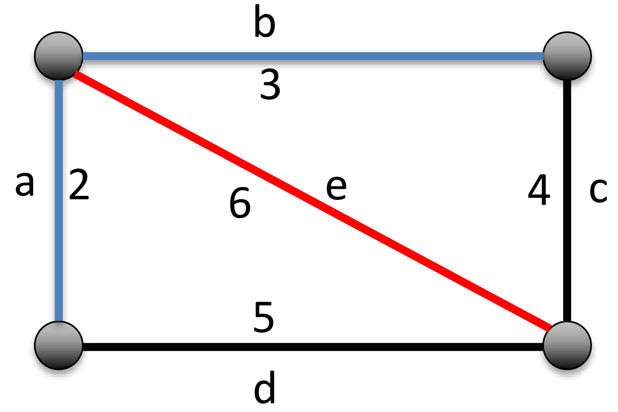
Balanced Thresholds – Cost and Remainder

Definition: Remainder(H,$w_1, \ldots, w_n$) = $\underset{S \subseteq OPT, S \cup H \text{ feasible}}{\operatorname{argmax}} \{\sum_{i \in S} w_i\}$.

- "Best subset of OPT that could have added to H."

Definition: Cost(H,$w_1, \ldots, w_n$) = OPT – Remainder(H).

- "What we lost from OPT by accepting H."



OPT = {e, d, c}.

Remainder({a}) = {e,c}. Cost({a}) = d.

Remainder({e}) = {d,c}. Cost({e}) =e.

Balanced Thresholds – Cost and Remainder

Definition: Remainder(H,$w_1, \ldots, w_n$) = $\underset{S \subseteq OPT, S \cup H \text{ feasible}}{\operatorname{argmax}} \{\sum_{i \in S} w_i\}$.

- "Best subset of OPT that could have added to H."

Definition: Cost(H,$w_1, \ldots, w_n$) = OPT – Remainder(H).

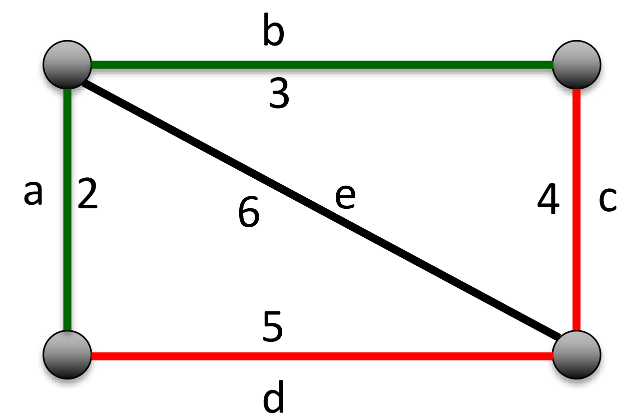
- "What we lost from OPT by accepting H."

OPT = {e, d, c}.

Remainder({a}) = {e,c}. Cost({a}) = d.

Remainder({e}) = {d,c}. Cost({e}) =e.

Remainder({a,b}) = {e}. Cost({a,b}) = {c,d}.



Balanced Thresholds

Definition: Remainder(H,$w_1, \dots, w_n$) = $\underset{S \subseteq OPT, S \cup H \text{ feasible}}{\operatorname{argmax}} \{\sum_{i \in S} w_i\}$.

- "Best subset of OPT that could have added to H."

Definition: Cost(H,$w_1, \dots, w_n$) = OPT – Remainder(H).

- "What we lost from OPT by accepting H."

Definition: A **thresholding algorithm** defines thresholds $T_i(w_1, \dots, w_{i-1})$, accepts i iff $w_i > T_i$ and feasible to hire i.

Will just write T_i, but remember can depend on $w_1, \dots, w_{i-1}$.

Definition: A thresholding algorithm has α-balanced thresholds if whenever it accepts set H when the weights are $w_1, \dots, w_n$, we have:

- Thresholds **not too small**: $\sum_{i \in H} T_i \geq \frac{1}{\alpha} E[\text{Cost}(H, \widehat{w}_1, \dots, \widehat{w}_n)]$.
- Thresholds **not too big**: $\sum_{i \in V} T_i \leq \left(1 - \frac{1}{\alpha}\right) E[\text{Remainder}(H, \widehat{w}_1, \dots, \widehat{w}_n)]$, for all V disjoint from H such that $H \cup V$ is feasible.
- $\widehat{w}_1, \dots, \widehat{w}_n$ denote fresh samples from $D_1, \dots, D_n$.

Expected Cost and Remainder

Definition: Remainder(H,$w_1, \dots, w_n$) = $\underset{S \subseteq OPT, S \cup H \text{ feasible}}{\operatorname{argmax}} \{\sum_{i \in S} w_i\}$.

- "Best subset of OPT that could have added to H."

Definition: Cost(H,$w_1, \dots, w_n$) = OPT – Remainder(H).

- "What we lost from OPT by accepting H."

Example: Sets of size 1 feasible. E[OPT] = 5/6.

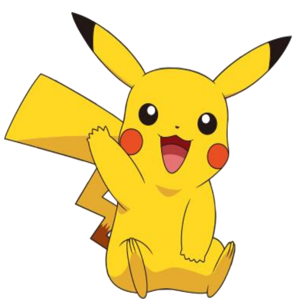
E[Remainder({Charmander})] = 0. E[Cost({Charmander})] = 5/6.

E[Remainder(∅)]= 5/6. E[Cost(∅)] = 0.

w = U[0,1]

U[0,1]

U[0,1]



U[0,1]

U[0,1]

Expected Cost and Remainder

Definition: Remainder(H,$w_1, \ldots, w_n$) = $\underset{S\subseteq OPT, S\cup H \text{ feasible}}{\operatorname{argmax}} \{\sum_{i\in S} w_i\}$.

- "Best subset of OPT that could have added to H."

Definition: Cost(H,$w_1, \ldots, w_n$) = OPT – Remainder(H).

- "What we lost from OPT by accepting H."

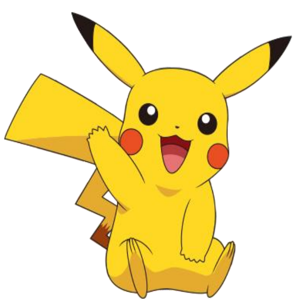
Example: Sets of size 2 feasible. E[OPT] = 5/6+2/3 = 3/2.

E[Remainder({Charmander})] = 5/6. E[Cost({Charmander})] = 2/3.

E[Remainder(Ø)] = 3/2. E[Cost(Ø)] = 0.

E[Remainder({Bulbasaur, Squirtle})] = 0. E[Cost({Bulbasaur, Squirtle})] = 3/2.




w = U[0,1] U[0,1] U[0,1] U[0,1] U[0,1]

Expected Cost and Remainder

Definition: Remainder(H,$w_1, \dots, w_n$) = $\underset{S \subseteq OPT, S \cup H \text{ feasible}}{\text{argmax}} \{\sum_{i \in S} w_i\}$.

- "Best subset of OPT that could have added to H."

Definition: Cost(H,$w_1, \dots, w_n$) = OPT – Remainder(H).

- "What we lost from OPT by accepting H."

Example: Sets of size k feasible. OPT = top k elements.

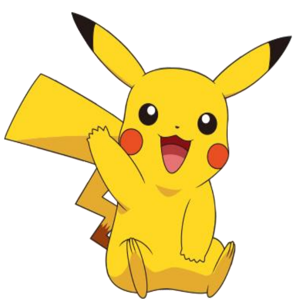
E[Remainder(H)] = Expected weight of top k-|H| elements.

E[Cost(H)] = Expected weight of lowest |H| elements of top k.

w = U[0,1]

U[0,1]

U[0,1]



U[0,1]

U[0,1]

Balanced Thresholds Imply Prophet Inequalities

Definition: A thresholding algorithm has α-balanced thresholds if whenever it accepts set H when the weights are $w_1, \ldots, w_n$, we have:

- Thresholds **not too small**: $\sum_{i \in H} T_i \geq \frac{1}{\alpha} E[\mathrm{Cost}(H, \widehat{w}_1, \ldots, \widehat{w}_n)]$.
- Thresholds **not too big**: $\sum_{i \in V} T_i \leq \left(1 - \frac{1}{\alpha}\right) E[\mathrm{Remainder}(H, \widehat{w}_1, \ldots, \widehat{w}_n)]$, for all V disjoint from H such that $H \cup V$ is feasible.

Theorem [KW 12]: If a thresholding algorithm has α-balanced thresholds, then it guarantees $E[ALG] \geq \frac{1}{\alpha} E[OPT]$.

Proof overview: Write E[OPT] = $E[\mathrm{Cost}(H, \widehat{w}_1, \ldots, \widehat{w}_n) + \mathrm{Remainder}(H, \widehat{w}, \ldots, \widehat{w}_n)]$.

Just partitions OPT($\widehat{w}_1, \ldots, \widehat{w}_n$) into Cost(H) and Remainder(H).

- "Not too small" guarantees $E[\sum_{i \in H} T_i] \geq 1/\alpha\, E[\mathrm{Cost}(H, \widehat{w}_1, \ldots, \widehat{w}_n)]$.
- "Not too big" guarantees $E[\sum_{i \in H} (w_i - T_i)] \geq 1/\alpha\, E[\mathrm{Remainder}(H, \widehat{w}_1, \ldots, \widehat{w}_n)]$.
- Summing yields $E[\sum_{i \in H} w_i] \geq E[OPT]/\alpha$.

Proving Thresholds are Balanced: 1-uniform

Definition: A thresholding algorithm has α-balanced thresholds if whenever it accepts set H when the weights are $w_1, \dots, w_n$, we have:

- Thresholds **not too small**: $\sum_{i \in H} T_i \geq \frac{1}{\alpha} E[\mathrm{Cost}(H, \widehat{w}_1, \dots, \widehat{w}_n)]$.
- Thresholds **not too big**: $\sum_{i \in V} T_i \leq \left(1 - \frac{1}{\alpha}\right) E[\mathrm{Remainder}(H, \widehat{w}_1, \dots, \widehat{w}_n)]$, for all V disjoint from H such that $H \cup V$ is feasible.

Theorem [KW 12]: If a thresholding algorithm has α-balanced thresholds, then it guarantees $E[ALG] \geq \frac{1}{\alpha} E[OPT]$.

Observation: For 1-uniform matroids, $T = E\left[\max_i \{w_i\}\right]/2$ are 2-balanced.

- Any hired element i has $E[\mathrm{Cost}(i)] = 2T$, so **not too small**.
- If nothing accepted, all possible V have |V| = 1, $E[\mathrm{Remainder}(\emptyset)] = 2T$.
- If something accepted, possible V = Ø, constraint becomes $0 \leq 0$. So **not too big**.

Proving Thresholds are Balanced: k-uniform

Theorem [(modified) CHMS 10]: 2-balanced thresholds exist for k-uniform matroids.

- Set $T_i = \frac{E[OPT]}{2k}$, for all i.

Proof:

- What is Remainder(H)? Highest weight k-|H| elements.
- What is Cost(H)? |H| lowest weight items in the top k.
- So $E[\text{Remainder}(H)] \geq \left(\frac{k-|H|}{k}\right) E[OPT]$.
- $E[Cost(H)] \leq \frac{|H|}{k} E[OPT]$.

- $\Rightarrow \sum_{i \in H} T_i = \frac{|H|}{2k} E[OPT] \geq E[\text{Cost}(H)]/2$, **not too small.**
- $\Rightarrow \sum_{i \in V} T_i \leq \frac{k-|H|}{2k} E[OPT] \leq E[\text{Remainder}(H)]/2$, **not too big.**

Proving *Adaptive* Thresholds are Balanced: k-uniform

Theorem [(modified) CHMS 10]: 2-balanced thresholds exist for k-uniform matroids.

- Set $T_i = \frac{E\left[OPT_{k-|H_{i-1}|}\right]}{2}$, for all i. H_{i-1} = hired secretaries from $\{1, \ldots, i-1\}$.
  - OPT_c = expected weight of c^{th} highest element.

Alternative Proof:

- What is Remainder(H)? Highest weight k-|H| elements.
- What is Cost(H)? |H| lowest weight items in the top k.
- So $E[\text{Remainder}(H)] = \sum_{c=1}^{k-|H|} E[OPT_c]$.
- $E[Cost(H)] = \sum_{c=0}^{|H|-1} E[OPT_{k-c}]$.

- $\Rightarrow \sum_{i \in H} T_i = \sum_{c=0}^{|H|-1} E[OPT_{k-c}]/2 = E[\text{Cost}(H)]/2$, **not too small**.
- $\Rightarrow \sum_{i \in V} T_i \leq (k - |H|) \cdot E\left[OPT_{k-|H|}\right]/2 \leq E[\text{Remainder}(H)]/2$, **not too big**.

Proving Thresholds are Balanced: Matroids

Theorem [KW 12]: 2-balanced thresholds exist for all matroids.

- Set $T_i = \frac{E[\text{Cost}(H_{i-1} \cup \{i\}, \widehat{w}_1, \ldots, \widehat{w}_n)] - E[\text{Cost}(H_{i-1}, \widehat{w}_1, \ldots, \widehat{w}_n)]}{2}$ for all i.

Omit proof. Intuition for thresholds – Imagine two worlds:

A: All weights redrawn fresh, game restarted, but already hired secretaries H_{i-1}.

B: All weights redrawn fresh, game restarted, but already hired secretaries $H_{i-1} \cup \{i\}$.

- Clearly, World A is better.
    - If you are a prophet, by exactly $E[\text{Cost}(H_{i-1} \cup \{i\}) - \text{Cost}(H_{i-1})]$.
- So in order to prefer World B, w_i should be $\Omega(E[\text{Cost}(H_{i-1} \cup \{i\}) - \text{Cost}(H_{i-1})])$.
    - Dividing by 2 just makes the math work out.

Recap - Balanced Thresholds

- **Not too small** = Thresholds themselves cover part of expected OPT.
- **Not too big** = Expected surplus above thresholds still large.

Another kind of balanced thresholds, by probability [**Samuel-Cahn 86, CHMS 10**]:

- **Not too small** = unlikely to block any element.
- **Not too big** = accept enough elements in expectation.
- Related to “contention resolution schemes” [**Feldman-Svensson-Zenklusen 16**].

Not all proofs follow this methodology, but it’s a good way to think about the “challenge” of prophet inequalities.

Related Results/Problems

What if you get to choose the order?

- Improve to e/(e-1) approximation for all matroids (tight) [**Yan 11**].

Algorithm:

- Compute $q_i = \Pr[i \in OPT]$ for all i.
- Set T_i such that $\Pr[w_i > T_i] = q_i$.
- Sort i in decreasing order of T_i.
- Hire every i with $w_i > T_i$, (and feasible to hire i).

Proof Overview: Uses “Correlation Gap Inequalities.”

Related Results/Problems

What if you have limited access to D_i?

- $1 + O(1/\sqrt{k})$ for k-uniform with 1 sample from each [**Azar-Kleinberg-W. 14**].
- Open: What is the best ratio for 1-uniform with 1 sample?
  - Set T = highest sample gets <4-approximation.
- Open: O(1) approximation for matroids with 1 sample from each?

What if an adversary adaptively chooses the ordering?

- Most results hold even if adversary “is a prophet” (knows weights).
  - Exception: [**KW 12**], holds if adversary “is a gambler” (knows what you know).

Applications to Bayesian Mechanism Design

- Good prophet inequalities against appropriate adversaries immediately imply good mechanisms in certain Bayesian settings [**CHMS 10**].
- See Anna’s talk on Friday for more details!

Related Results/Problems

Thanks for listening!