Topics in Algorithmic Game Theory and Economics

Pieter Kleer

Max Planck Institute for Informatics (D1)
Saarland Informatics Campus

February 3, 2020

Lecture 11
Prophet Inequalities

Online selection

Consider

- Finite set of elements $E = \{e_1, \ldots, e_m\}$.

Online selection

Consider

- Finite set of elements $E = \{e_1, \ldots, e_m\}$.
- Weight function $w : E \to \mathbb{R}_{\geq 0}$.

Online selection

Consider

- Finite set of elements $E = \{e_1, \ldots, e_m\}$.
- Weight function $w : E \to \mathbb{R}_{\geq 0}$.
- Collection of feasible subsets $\mathcal{F} \subseteq 2^E = \{S : S \subseteq E\}$.

Online selection

Consider

- Finite set of elements $E = \{e_1, \ldots, e_m\}$.
- Weight function $w : E \to \mathbb{R}_{\geq 0}$.
- Collection of feasible subsets $\mathcal{F} \subseteq 2^E = \{S : S \subseteq E\}$.

Elements arrive one by one in unknown order $\sigma = (\sigma(1), \ldots, \sigma(m))$.

Online selection

Consider

- Finite set of elements $E = \{e_1, \ldots, e_m\}$.
- Weight function $w : E \to \mathbb{R}_{\geq 0}$.
- Collection of feasible subsets $\mathcal{F} \subseteq 2^E = \{S : S \subseteq E\}$.

Elements arrive one by one in unknown order $\sigma = (\sigma(1), \ldots, \sigma(m))$.

Online selection problem with initial $S = \emptyset$

Online selection

Consider

- Finite set of elements $E = \{e_1, \ldots, e_m\}$.
- Weight function $w : E \to \mathbb{R}_{\geq 0}$.
- Collection of feasible subsets $\mathcal{F} \subseteq 2^E = \{S : S \subseteq E\}$.

Elements arrive one by one in unknown order $\sigma = (\sigma(1), \ldots, \sigma(m))$.

Online selection problem with initial $S = \emptyset$

Online selection

Consider

- Finite set of elements $E = \{e_1, \ldots, e_m\}$.
- Weight function $w : E \to \mathbb{R}_{\geq 0}$.
- Collection of feasible subsets $\mathcal{F} \subseteq 2^E = \{S : S \subseteq E\}$.

Elements arrive one by one in unknown order $\sigma = (\sigma(1), \ldots, \sigma(m))$.

Online selection problem with initial $S = \emptyset$

For $i = 1, \ldots, m$, upon arrival of element $\sigma(i)$:

Online selection

Consider

- Finite set of elements $E = \{e_1, \ldots, e_m\}$.
- Weight function $w : E \to \mathbb{R}_{\geq 0}$.
- Collection of feasible subsets $\mathcal{F} \subseteq 2^E = \{S : S \subseteq E\}$.

Elements arrive one by one in unknown order $\sigma = (\sigma(1), \ldots, \sigma(m))$.

Online selection problem with initial $S = \emptyset$

For $i = 1, \ldots, m$, upon arrival of element $\sigma(i)$:

- Weight $w_{\sigma(i)}$ is revealed.

Online selection

Consider

- Finite set of elements $E = \{e_1, \ldots, e_m\}$.
- Weight function $w : E \to \mathbb{R}_{\geq 0}$.
- Collection of feasible subsets $\mathcal{F} \subseteq 2^E = \{S : S \subseteq E\}$.

Elements arrive one by one in unknown order $\sigma = (\sigma(1), \ldots, \sigma(m))$.

Online selection problem with initial $S = \emptyset$

For $i = 1, \ldots, m$, upon arrival of element $\sigma(i)$:

- Weight $w_{\sigma(i)}$ is revealed.
- Decide (irrevocably) whether to select or reject $\sigma(i)$,

Online selection

Consider

- Finite set of elements $E = \{e_1, \ldots, e_m\}$.
- Weight function $w : E \to \mathbb{R}_{\geq 0}$.
- Collection of feasible subsets $\mathcal{F} \subseteq 2^E = \{S : S \subseteq E\}$.

Elements arrive one by one in unknown order $\sigma = (\sigma(1), \ldots, \sigma(m))$.

Online selection problem with initial $S = \emptyset$

For $i = 1, \ldots, m$, upon arrival of element $\sigma(i)$:

- Weight $w_{\sigma(i)}$ is revealed.
- Decide (irrevocably) whether to select or reject $\sigma(i)$, where selecting is only allowed if $S + \sigma(i) \in \mathcal{F}$.

Online selection

Consider

- Finite set of elements $E = \{e_1, \ldots, e_m\}$.
- Weight function $w : E \to \mathbb{R}_{\geq 0}$.
- Collection of feasible subsets $\mathcal{F} \subseteq 2^E = \{S : S \subseteq E\}$.

Elements arrive one by one in unknown order $\sigma = (\sigma(1), \ldots, \sigma(m))$.

Online selection problem with initial $S = \emptyset$

For $i = 1, \ldots, m$, upon arrival of element $\sigma(i)$:

- Weight $w_{\sigma(i)}$ is revealed.
- Decide (irrevocably) whether to select or reject $\sigma(i)$, where selecting is only allowed if $S + \sigma(i) \in \mathcal{F}$.

Goal: *Select subset $S \in \mathcal{F}$ maximizing $w(S) = \sum_{e \in S} w(e)$.*

Bayesian setting

With adversarial arrival order

Bayesian setting

Instead of making assumption on arrival order (uniform random), we make assumption on the (unknown) weights of the elements.

Bayesian setting

Instead of making assumption on arrival order (uniform random), we make assumption on the (unknown) weights of the elements.

In Bayesian setting, we have for every element i a (non-negative) probability distribution $X_i : \mathbb{R}_{\geq 0} \rightarrow [0, 1]$.

Bayesian setting

Instead of making assumption on arrival order (uniform random), we make assumption on the (unknown) weights of the elements.

In Bayesian setting, we have for every element i a (non-negative) probability distribution $X_i : \mathbb{R}_{\geq 0} \rightarrow [0, 1]$.

- Distributions X_i are independent from each other.

Bayesian setting

Instead of making assumption on arrival order (uniform random), we make assumption on the (unknown) weights of the elements.

In Bayesian setting, we have for every element i a (non-negative) probability distribution $X_i : \mathbb{R}_{\geq 0} \to [0, 1]$.

- Distributions X_i are independent from each other.
- Weight w_i of element e_i is sample from X_i.

Bayesian setting

Instead of making assumption on arrival order (uniform random), we make assumption on the (unknown) weights of the elements.

In Bayesian setting, we have for every element i a (non-negative) probability distribution $X_i : \mathbb{R}_{\geq 0} \to [0, 1]$.

- Distributions X_i are independent from each other.
- Weight w_i of element e_i is sample from X_i.

Online procedure for set system $\mathcal{F} = (E, \mathcal{I})$:
Set $S = \emptyset$.

Bayesian setting

Instead of making assumption on arrival order (uniform random), we make assumption on the (unknown) weights of the elements.

In Bayesian setting, we have for every element i a (non-negative) probability distribution $X_i : \mathbb{R}_{\geq 0} \to [0, 1]$.

- Distributions X_i are independent from each other.
- Weight w_i of element e_i is sample from X_i.

Online procedure for set system $\mathcal{F} = (E, \mathcal{I})$:
Set $S = \emptyset$.

- For every i, a realization $w_i \sim X_i$ is generated.

Bayesian setting

Instead of making assumption on arrival order (uniform random), we make assumption on the (unknown) weights of the elements.

In Bayesian setting, we have for every element i a (non-negative) probability distribution $X_i : \mathbb{R}_{\geq 0} \rightarrow [0, 1]$.

- Distributions X_i are independent from each other.
- Weight w_i of element e_i is sample from X_i.

Online procedure for set system $\mathcal{F} = (E, \mathcal{I})$:
Set $S = \emptyset$.

- For every i, a realization $w_i \sim X_i$ is generated.
    - All realizations w_i are shown to the adversary.

Bayesian setting

Instead of making assumption on arrival order (uniform random), we make assumption on the (unknown) weights of the elements.

In Bayesian setting, we have for every element i a (non-negative) probability distribution $X_i : \mathbb{R}_{\geq 0} \rightarrow [0, 1]$.

- Distributions X_i are independent from each other.
- Weight w_i of element e_i is sample from X_i.

Online procedure for set system $\mathcal{F} = (E, \mathcal{I})$:
Set $S = \emptyset$.

- For every i, a realization $w_i \sim X_i$ is generated.
  - All realizations w_i are shown to the adversary.
- For $i = 1, \ldots, m$:

Bayesian setting

Instead of making assumption on arrival order (uniform random), we make assumption on the (unknown) weights of the elements.

In Bayesian setting, we have for every element i a (non-negative) probability distribution $X_i : \mathbb{R}_{\geq 0} \rightarrow [0, 1]$.

- Distributions X_i are independent from each other.
- Weight w_i of element e_i is sample from X_i.

Online procedure for set system $\mathcal{F} = (E, \mathcal{I})$:
Set $S = \emptyset$.

- For every i, a realization $w_i \sim X_i$ is generated.
  - All realizations w_i are shown to the adversary.
- For $i = 1, \ldots, m$:
  - Adversary chooses $\sigma(i) \in E$, and reveals it and its weight w_i.

Bayesian setting

Instead of making assumption on arrival order (uniform random), we make assumption on the (unknown) weights of the elements.

In Bayesian setting, we have for every element i a (non-negative) probability distribution $X_i : \mathbb{R}_{\geq 0} \rightarrow [0, 1]$.

- Distributions X_i are independent from each other.
- Weight w_i of element e_i is sample from X_i.

Online procedure for set system $\mathcal{F} = (E, \mathcal{I})$:
Set $S = \emptyset$.

- For every i, a realization $w_i \sim X_i$ is generated.
    - All realizations w_i are shown to the adversary.
- For $i = 1, \ldots, m$:
    - Adversary chooses $\sigma(i) \in E$, and reveals it and its weight w_i.
    - Online algorithm $\mathcal{A}$ decides whether to accept or reject $\sigma(i)$,

Bayesian setting

Instead of making assumption on arrival order (uniform random), we make assumption on the (unknown) weights of the elements.

In Bayesian setting, we have for every element i a (non-negative) probability distribution $X_i : \mathbb{R}_{\geq 0} \rightarrow [0, 1]$.

- Distributions X_i are independent from each other.
- Weight w_i of element e_i is sample from X_i.

Online procedure for set system $\mathcal{F} = (E, \mathcal{I})$:
Set $S = \emptyset$.

- For every i, a realization $w_i \sim X_i$ is generated.
  - All realizations w_i are shown to the adversary.
- For $i = 1, \ldots, m$:
  - Adversary chooses $\sigma(i) \in E$, and reveals it and its weight w_i.
  - Online algorithm $\mathcal{A}$ decides whether to accept or reject $\sigma(i)$, where acceptance is only allowed if $S + \sigma(i) \in \mathcal{F}$.

Probability distributions

Very roughly speaking, there are two main types of probability distributions: *continuous* and *discrete*.

Probability distributions

Very roughly speaking, there are two main types of probability distributions: *continuous* and *discrete*.

- A non-negative discrete random variable X is given by function $g : \mathbb{N} \to [0, 1]$ with

$$\sum_{i=0}^{\infty} g(i) = 1.$$

Probability distributions

Very roughly speaking, there are two main types of probability distributions: *continuous* and *discrete*.

- A non-negative discrete random variable X is given by function $g : \mathbb{N} \to [0, 1]$ with

$$\sum_{i=0}^{\infty} g(i) = 1.$$

Example

Suppose we have a fair die with six sides. Then $g(i) = \frac{1}{6}$ for $i = 1, \ldots, 6$ and $g(i) = 0$ otherwise.

Probability distributions

Very roughly speaking, there are two main types of probability distributions: *continuous* and *discrete*.

- A non-negative discrete random variable X is given by function $g : \mathbb{N} \to [0, 1]$ with

$$\sum_{i=0}^{\infty} g(i) = 1.$$

Example

Suppose we have a fair die with six sides. Then $g(i) = \frac{1}{6}$ for $i = 1, \ldots, 6$ and $g(i) = 0$ otherwise.

- A non-negative continuous random variable X is given by density function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ with

$$\int_0^{\infty} f(x)dx = 1.$$

Probability distributions

Very roughly speaking, there are two main types of probability distributions: *continuous* and *discrete*.

- A non-negative discrete random variable X is given by function $g : \mathbb{N} \to [0, 1]$ with

$$\sum_{i=0}^{\infty} g(i) = 1.$$

Example

Suppose we have a fair die with six sides. Then $g(i) = \frac{1}{6}$ for $i = 1, \ldots, 6$ and $g(i) = 0$ otherwise.

- A non-negative continuous random variable X is given by density function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ with

$$\int_0^{\infty} f(x)dx = 1.$$

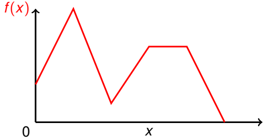
  It then holds that $\mathbb{P}(X \leq z) = \int_0^z f(x)dx$.

$$\mathbb{P}(X \leq z) = \int_0^z f(x)dx$$

The function $f(x)$ models how the probability mass is spread out.

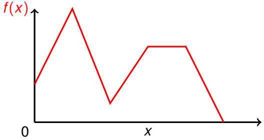
$$\mathbb{P}(X \leq z) = \int_0^z f(x)dx$$

The function $f(x)$ models how the probability mass is spread out.



$$\mathbb{P}(X \leq z) = \int_0^z f(x)dx$$

The function $f(x)$ models how the probability mass is spread out.

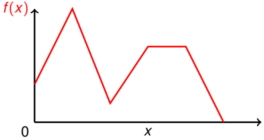


Example

Consider the uniform distribution over the interval $[a, b]$ with $0 \leq a < b$. Then $f(x) = \frac{1}{b-a}$.

$$\mathbb{P}(X \le z) = \int_0^z f(x)dx$$

The function $f(x)$ models how the probability mass is spread out.



Example

Consider the uniform distribution over the interval $[a, b]$ with $0 \le a < b$. Then $f(x) = \frac{1}{b-a}$.

Remark

All the results we discuss today hold for both continuous and discrete distributions, but sometimes need slightly different arguments.

Online procedure for set system $\mathcal{F} = (E, \mathcal{I})$:

Set $S = \emptyset$.

- For every i, a realization $w_i \sim X_i$ is generated.
  - All realizations w_i are shown to the adversary.
- For $i = 1, \ldots, m$:
  - Adversary chooses $\sigma(i) \in E$, and reveals it and its weight w_i.
  - Online algorithm $\mathcal{A}$ decides whether to accept or reject $\sigma(i)$, where acceptance is only allowed if $S + \sigma(i) \in \mathcal{F}$.

Online procedure for set system $\mathcal{F} = (E, \mathcal{I})$:

Set $S = \emptyset$.

- For every i, a realization $w_i \sim X_i$ is generated.
  - All realizations w_i are shown to the adversary.
- For $i = 1, \ldots, m$:
  - Adversary chooses $\sigma(i) \in E$, and reveals it and its weight w_i.
  - Online algorithm $\mathcal{A}$ decides whether to accept or reject $\sigma(i)$, where acceptance is only allowed if $S + \sigma(i) \in \mathcal{F}$.

- Algorithm $\mathcal{A}$ may use (in step i) information revealed so far, as well as the distributions X_i of all elements.

Online procedure for set system $\mathcal{F} = (E, \mathcal{I})$:
Set $S = \emptyset$.

- For every i, a realization $w_i \sim X_i$ is generated.
  - All realizations w_i are shown to the adversary.
- For $i = 1, \ldots, m$:
  - Adversary chooses $\sigma(i) \in E$, and reveals it and its weight w_i.
  - Online algorithm $\mathcal{A}$ decides whether to accept or reject $\sigma(i)$, where acceptance is only allowed if $S + \sigma(i) \in \mathcal{F}$.

- Algorithm $\mathcal{A}$ may use (in step i) information revealed so far, as well as the distributions X_i of all elements.

About the adversary

Online procedure for set system $\mathcal{F} = (E, \mathcal{I})$:
Set $S = \emptyset$.

- For every i, a realization $w_i \sim X_i$ is generated.
  - All realizations w_i are shown to the adversary.
- For $i = 1, \ldots, m$:
  - Adversary chooses $\sigma(i) \in E$, and reveals it and its weight w_i.
  - Online algorithm $\mathcal{A}$ decides whether to accept or reject $\sigma(i)$, where acceptance is only allowed if $S + \sigma(i) \in \mathcal{F}$.

- Algorithm $\mathcal{A}$ may use (in step i) information revealed so far, as well as the distributions X_i of all elements.

About the adversary

In general, we assume to have an all-knowing, adaptive adversary

Online procedure for set system $\mathcal{F} = (E, \mathcal{I})$:
Set $S = \emptyset$.

- For every i, a realization $w_i \sim X_i$ is generated.
  - All realizations w_i are shown to the adversary.
- For $i = 1, \ldots, m$:
  - Adversary chooses $\sigma(i) \in E$, and reveals it and its weight w_i.
  - Online algorithm $\mathcal{A}$ decides whether to accept or reject $\sigma(i)$, where acceptance is only allowed if $S + \sigma(i) \in \mathcal{F}$.

- Algorithm $\mathcal{A}$ may use (in step i) information revealed so far, as well as the distributions X_i of all elements.

About the adversary

In general, we assume to have an all-knowing, adaptive adversary

- Can choose which element to present in step i, based on

Online procedure for set system $\mathcal{F} = (E, \mathcal{I})$:
Set $S = \emptyset$.

- For every i, a realization $w_i \sim X_i$ is generated.
  - All realizations w_i are shown to the adversary.
- For $i = 1, \ldots, m$:
  - Adversary chooses $\sigma(i) \in E$, and reveals it and its weight w_i.
  - Online algorithm $\mathcal{A}$ decides whether to accept or reject $\sigma(i)$, where acceptance is only allowed if $S + \sigma(i) \in \mathcal{F}$.

- Algorithm $\mathcal{A}$ may use (in step i) information revealed so far, as well as the distributions X_i of all elements.

About the adversary

In general, we assume to have an all-knowing, adaptive adversary

- Can choose which element to present in step i, based on
  - Choices of online algorithm in steps $1, \ldots, i-1$.

Online procedure for set system $\mathcal{F} = (E, \mathcal{I})$:
Set $S = \emptyset$.

- For every i, a realization $w_i \sim X_i$ is generated.
  - All realizations w_i are shown to the adversary.
- For $i = 1, \ldots, m$:
  - Adversary chooses $\sigma(i) \in E$, and reveals it and its weight w_i.
  - Online algorithm $\mathcal{A}$ decides whether to accept or reject $\sigma(i)$, where acceptance is only allowed if $S + \sigma(i) \in \mathcal{F}$.

- Algorithm $\mathcal{A}$ may use (in step i) information revealed so far, as well as the distributions X_i of all elements.

About the adversary

In general, we assume to have an all-knowing, adaptive adversary

- Can choose which element to present in step i, based on
  - Choices of online algorithm in steps $1, \ldots, i-1$.
  - Realizations of all elements (including those that have not arrived).

Online procedure for set system $\mathcal{F} = (E, \mathcal{I})$:
Set $S = \emptyset$.

- For every i, a realization $w_i \sim X_i$ is generated.
  - All realizations w_i are shown to the adversary.
- For $i = 1, \dots, m$:
  - Adversary chooses $\sigma(i) \in E$, and reveals it and its weight w_i.
  - Online algorithm $\mathcal{A}$ decides whether to accept or reject $\sigma(i)$, where acceptance is only allowed if $S + \sigma(i) \in \mathcal{F}$.

- Algorithm $\mathcal{A}$ may use (in step i) information revealed so far, as well as the distributions X_i of all elements.

About the adversary

In general, we assume to have an all-knowing, adaptive adversary

- Can choose which element to present in step i, based on
  - Choices of online algorithm in steps $1, \dots, i-1$.
  - Realizations of all elements (including those that have not arrived).

Adversary is non-adaptive if order is fixed after seeing all realizations.

Example

Let $E = \{e_1, e_2\}$ of which we may select at most one element.

Example

Let $E = \{e_1, e_2\}$ of which we may select at most one element.
Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$.

Example

Let $E = \{e_1, e_2\}$ of which we may select at most one element.
Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$. Distributions are given by:

$$w_1 \sim X_1 = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 0 & \text{with probability } 1 - \epsilon \end{cases} \tag{1}$$

Example

Let $E = \{e_1, e_2\}$ of which we may select at most one element.
Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$. Distributions are given by:

$$w_1 \sim X_1 = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 0 & \text{with probability } 1 - \epsilon \end{cases} \tag{1}$$

$$w_2 \sim X_2 = \begin{cases} 1 + \delta & \text{with probability } 1. \end{cases} \tag{2}$$

Example

Let $E = \{e_1, e_2\}$ of which we may select at most one element.
Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$. Distributions are given by:

$$w_1 \sim X_1 = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 0 & \text{with probability } 1 - \epsilon \end{cases} \tag{1}$$

$$w_2 \sim X_2 = \left\{ 1 + \delta \quad \text{with probability } 1. \right. \tag{2}$$

Note that $\mathbb{E}[X_1] = \frac{1}{\epsilon} \times \epsilon + 0 \times (1 - \epsilon) = 1$ and $\mathbb{E}[X_2] = 1 + \delta$.

Example

Let $E = \{e_1, e_2\}$ of which we may select at most one element.
Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$. Distributions are given by:

$$w_1 \sim X_1 = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 0 & \text{with probability } 1 - \epsilon \end{cases} \quad (1)$$

$$w_2 \sim X_2 = \left\{ \begin{array}{ll} 1 + \delta & \text{with probability } 1. \end{array} \right. \quad (2)$$

Note that $\mathbb{E}[X_1] = \frac{1}{\epsilon} \times \epsilon + 0 \times (1 - \epsilon) = 1$ and $\mathbb{E}[X_2] = 1 + \delta$.

- If arrival order would be (e_1, e_2), simply observe realization w_1.

Example

Let $E = \{e_1, e_2\}$ of which we may select at most one element.
Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$. Distributions are given by:

$$w_1 \sim X_1 = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 0 & \text{with probability } 1 - \epsilon \end{cases} \tag{1}$$

$$w_2 \sim X_2 = \{ \, 1 + \delta \quad \text{with probability } 1. \tag{2}$$

Note that $\mathbb{E}[X_1] = \frac{1}{\epsilon} \times \epsilon + 0 \times (1 - \epsilon) = 1$ and $\mathbb{E}[X_2] = 1 + \delta$.

- If arrival order would be (e_1, e_2), simply observe realization w_1.
  - If $w_1 = 1/\epsilon$, then select e_1 (as $\frac{1}{\epsilon} > 1 + \delta$).

Example

Let $E = \{e_1, e_2\}$ of which we may select at most one element.
Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$. Distributions are given by:

$$w_1 \sim X_1 = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 0 & \text{with probability } 1 - \epsilon \end{cases} \quad (1)$$

$$w_2 \sim X_2 = \left\{ 1 + \delta \quad \text{with probability } 1. \right. \quad (2)$$

Note that $\mathbb{E}[X_1] = \frac{1}{\epsilon} \times \epsilon + 0 \times (1 - \epsilon) = 1$ and $\mathbb{E}[X_2] = 1 + \delta$.

- If arrival order would be (e_1, e_2), simply observe realization w_1.
  - If $w_1 = 1/\epsilon$, then select e_1 (as $\frac{1}{\epsilon} > 1 + \delta$).
  - If $w_1 = 0$, reject e_1 and select e_2.

Example

Let $E = \{e_1, e_2\}$ of which we may select at most one element.
Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$. Distributions are given by:

$$w_1 \sim X_1 = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 0 & \text{with probability } 1 - \epsilon \end{cases} \quad (1)$$

$$w_2 \sim X_2 = \{ 1 + \delta \quad \text{with probability } 1. \quad (2)$$

Note that $\mathbb{E}[X_1] = \frac{1}{\epsilon} \times \epsilon + 0 \times (1 - \epsilon) = 1$ and $\mathbb{E}[X_2] = 1 + \delta$.

- If arrival order would be (e_1, e_2), simply observe realization w_1.
  - If $w_1 = 1/\epsilon$, then select e_1 (as $\frac{1}{\epsilon} > 1 + \delta$).
  - If $w_1 = 0$, reject e_1 and select e_2.
- Worst-case arrival order is (e_2, e_1).

Example

Let $E = \{e_1, e_2\}$ of which we may select at most one element.
Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$. Distributions are given by:

$$w_1 \sim X_1 = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 0 & \text{with probability } 1 - \epsilon \end{cases} \quad (1)$$

$$w_2 \sim X_2 = \{ \; 1 + \delta \quad \text{with probability } 1. \quad (2)$$

Note that $\mathbb{E}[X_1] = \frac{1}{\epsilon} \times \epsilon + 0 \times (1 - \epsilon) = 1$ and $\mathbb{E}[X_2] = 1 + \delta$.

- If arrival order would be (e_1, e_2), simply observe realization w_1.
  - If $w_1 = 1/\epsilon$, then select e_1 (as $\frac{1}{\epsilon} > 1 + \delta$).
  - If $w_1 = 0$, reject e_1 and select e_2.
- Worst-case arrival order is (e_2, e_1).
  - We don't know realization w_1, when deciding on element e_2.

Example

Let $E = \{e_1, e_2\}$ of which we may select at most one element.
Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$. Distributions are given by:

$$w_1 \sim X_1 = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 0 & \text{with probability } 1 - \epsilon \end{cases} \quad (1)$$

$$w_2 \sim X_2 = \{\ 1 + \delta \quad \text{with probability } 1. \quad (2)$$

Note that $\mathbb{E}[X_1] = \frac{1}{\epsilon} \times \epsilon + 0 \times (1 - \epsilon) = 1$ and $\mathbb{E}[X_2] = 1 + \delta$.

- If arrival order would be (e_1, e_2), simply observe realization w_1.
  - If $w_1 = 1/\epsilon$, then select e_1 (as $\frac{1}{\epsilon} > 1 + \delta$).
  - If $w_1 = 0$, reject e_1 and select e_2.
- Worst-case arrival order is (e_2, e_1).
  - We don't know realization w_1, when deciding on element e_2.
  - Nevertheless, it is (intuitively) optimal to select e_2.

Example

Let $E = \{e_1, e_2\}$ of which we may select at most one element.
Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$. Distributions are given by:

$$w_1 \sim X_1 = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 0 & \text{with probability } 1 - \epsilon \end{cases} \quad (1)$$

$$w_2 \sim X_2 = \{ 1 + \delta \quad \text{with probability } 1. \quad (2)$$

Note that $\mathbb{E}[X_1] = \frac{1}{\epsilon} \times \epsilon + 0 \times (1 - \epsilon) = 1$ and $\mathbb{E}[X_2] = 1 + \delta$.

- If arrival order would be (e_1, e_2), simply observe realization w_1.
  - If $w_1 = 1/\epsilon$, then select e_1 (as $\frac{1}{\epsilon} > 1 + \delta$).
  - If $w_1 = 0$, reject e_1 and select e_2.
- Worst-case arrival order is (e_2, e_1).
  - We don't know realization w_1, when deciding on element e_2.
  - Nevertheless, it is (intuitively) optimal to select e_2.
  - Why?

Example

Let $E = \{e_1, e_2\}$ of which we may select at most one element.
Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$. Distributions are given by:

$$w_1 \sim X_1 = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 0 & \text{with probability } 1 - \epsilon \end{cases} \quad (1)$$

$$w_2 \sim X_2 = \{ \, 1 + \delta \quad \text{with probability } 1. \quad (2)$$

Note that $\mathbb{E}[X_1] = \frac{1}{\epsilon} \times \epsilon + 0 \times (1 - \epsilon) = 1$ and $\mathbb{E}[X_2] = 1 + \delta$.

- If arrival order would be (e_1, e_2), simply observe realization w_1.
  - If $w_1 = 1/\epsilon$, then select e_1 (as $\frac{1}{\epsilon} > 1 + \delta$).
  - If $w_1 = 0$, reject e_1 and select e_2.
- Worst-case arrival order is (e_2, e_1).
  - We don't know realization w_1, when deciding on element e_2.
  - Nevertheless, it is (intuitively) optimal to select e_2.
  - Why? Deterministic value $w_2 = 1 + \delta > \mathbb{E}[X_1]$.

Example

Let $E = \{e_1, e_2\}$ of which we may select at most one element.
Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$. Distributions are given by:

$$w_1 \sim X_1 = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 0 & \text{with probability } 1 - \epsilon \end{cases} \quad (1)$$

$$w_2 \sim X_2 = \{\ 1 + \delta \quad \text{with probability } 1. \quad (2)$$

Note that $\mathbb{E}[X_1] = \frac{1}{\epsilon} \times \epsilon + 0 \times (1 - \epsilon) = 1$ and $\mathbb{E}[X_2] = 1 + \delta$.

- If arrival order would be (e_1, e_2), simply observe realization w_1.
  - If $w_1 = 1/\epsilon$, then select e_1 (as $\frac{1}{\epsilon} > 1 + \delta$).
  - If $w_1 = 0$, reject e_1 and select e_2.
- Worst-case arrival order is (e_2, e_1).
  - We don't know realization w_1, when deciding on element e_2.
  - Nevertheless, it is (intuitively) optimal to select e_2.
  - Why? Deterministic value $w_2 = 1 + \delta > \mathbb{E}[X_1]$.
    - In expectation (of X_1), we cannot do better if we reject e_2.

Example

Let $E = \{e_1, e_2\}$ of which we may select at most one element.
Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$. Distributions are given by:

$$w_1 \sim X_1 = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 0 & \text{with probability } 1 - \epsilon \end{cases} \tag{1}$$

$$w_2 \sim X_2 = \left\{ 1 + \delta \quad \text{with probability } 1. \right. \tag{2}$$

Note that $\mathbb{E}[X_1] = \frac{1}{\epsilon} \times \epsilon + 0 \times (1 - \epsilon) = 1$ and $\mathbb{E}[X_2] = 1 + \delta$.

- If arrival order would be (e_1, e_2), simply observe realization w_1.
  - If $w_1 = 1/\epsilon$, then select e_1 (as $\frac{1}{\epsilon} > 1 + \delta$).
  - If $w_1 = 0$, reject e_1 and select e_2.
- Worst-case arrival order is (e_2, e_1).
  - We don't know realization w_1, when deciding on element e_2.
  - Nevertheless, it is (intuitively) optimal to select e_2.
  - Why? Deterministic value $w_2 = 1 + \delta > \mathbb{E}[X_1]$.
    - In expectation (of X_1), we cannot do better if we reject e_2.

- Performance objective is formalized next.

Performance of online algorithm

Performance is measured against that of the prophet.

Performance of online algorithm

Performance is measured against that of the prophet.

- Prophet gets to see all realizations $w_i \sim X_i$ after they are sampled.

Performance of online algorithm

Performance is measured against that of the prophet.

- Prophet gets to see all realizations $w_i \sim X_i$ after they are sampled.
- Computes (offline) subset S^* with max. weight

$$\mathrm{OPT}(w_1, \ldots, w_m) := w(S^*) = max_{S \in \mathcal{F}} \sum_{e \in S} w_e.$$

Performance of online algorithm

Performance is measured against that of the prophet.

- Prophet gets to see all realizations $w_i \sim X_i$ after they are sampled.
- Computes (offline) subset S^* with max. weight

$$\mathrm{OPT}(w_1, \ldots, w_m) := w(S^*) = max_{S \in \mathcal{F}} \sum_{e \in S} w_e.$$

- Expected weight for prophet is

Performance of online algorithm

Performance is measured against that of the prophet.

- Prophet gets to see all realizations $w_i \sim X_i$ after they are sampled.
- Computes (offline) subset S^* with max. weight

$$\mathrm{OPT}(w_1, \ldots, w_m) := w(S^*) = max_{S \in \mathcal{F}} \sum_{e \in S} w_e.$$

- Expected weight for prophet is

$$\mathbf{OPT} = \mathbb{E}_{(\mathbf{y_1},\ldots,\mathbf{y_m}) \sim \mathbf{X_1} \times \cdots \times \mathbf{X_m}} \left[\mathrm{OPT}(\mathbf{y_1}, \ldots, \mathbf{y_m}) \right].$$

Performance of online algorithm

Performance is measured against that of the prophet.

- Prophet gets to see all realizations $w_i \sim X_i$ after they are sampled.
- Computes (offline) subset S^* with max. weight

$$\mathrm{OPT}(w_1, \ldots, w_m) := w(S^*) = max_{S \in \mathcal{F}} \sum_{e \in S} w_e.$$

- Expected weight for prophet is

$$\mathbf{OPT} = \mathbb{E}_{(\mathbf{y_1}, \ldots, \mathbf{y_m}) \sim \mathbf{X_1} \times \cdots \times \mathbf{X_m}} \left[\mathrm{OPT}(\mathbf{y_1}, \ldots, \mathbf{y_m}) \right].$$

Expected weight of (deterministic) algorithm $\mathcal{A}$ is

Performance of online algorithm

Performance is measured against that of the prophet.

- Prophet gets to see all realizations $w_i \sim X_i$ after they are sampled.
- Computes (offline) subset S^* with max. weight

$$\mathrm{OPT}(w_1, \dots, w_m) := w(S^*) = max_{S \in \mathcal{F}} \sum_{e \in S} w_e.$$

- Expected weight for prophet is

$$\mathbf{OPT} = \mathbb{E}_{(\mathbf{y_1}, \dots, \mathbf{y_m}) \sim \mathbf{X_1} \times \dots \times \mathbf{X_m}} \left[\mathrm{OPT}(\mathbf{y_1}, \dots, \mathbf{y_m}) \right].$$

Expected weight of (deterministic) algorithm $\mathcal{A}$ is

$$\mathbf{ALG} = \mathbb{E}_{(\mathbf{y_1}, \dots, \mathbf{y_m}) \sim \mathbf{X_1} \times \dots \times \mathbf{X_m}} \left[\min_\sigma \mathbf{w}(\mathcal{A}(\sigma, \mathbf{y_1}, \dots, \mathbf{y_m})) \right].$$

Performance of online algorithm

Performance is measured against that of the prophet.

- Prophet gets to see all realizations $w_i \sim X_i$ after they are sampled.
- Computes (offline) subset S^* with max. weight

$$\mathrm{OPT}(w_1, \ldots, w_m) := w(S^*) = max_{S \in \mathcal{F}} \sum_{e \in S} w_e.$$

- Expected weight for prophet is

$$\mathbf{OPT} = \mathbb{E}_{(\mathbf{y_1}, \ldots, \mathbf{y_m}) \sim \mathbf{X_1} \times \cdots \times \mathbf{X_m}} \left[\mathrm{OPT}(\mathbf{y_1}, \ldots, \mathbf{y_m}) \right].$$

Expected weight of (deterministic) algorithm $\mathcal{A}$ is

$$\mathbf{ALG} = \mathbb{E}_{(\mathbf{y_1}, \ldots, \mathbf{y_m}) \sim \mathbf{X_1} \times \cdots \times \mathbf{X_m}} \left[\min_\sigma \mathbf{w}(\mathcal{A}(\sigma, \mathbf{y_1}, \ldots, \mathbf{y_m})) \right].$$

- With $w(\mathcal{A}(\sigma, y_1, \ldots, y_m))$ weight of set outputted by $\mathcal{A}$.

Performance of online algorithm

Performance is measured against that of the prophet.

- Prophet gets to see all realizations $w_i \sim X_i$ after they are sampled.
- Computes (offline) subset S^* with max. weight

$$\text{OPT}(w_1, \dots, w_m) := w(S^*) = max_{S \in \mathcal{F}} \sum_{e \in S} w_e.$$

- Expected weight for prophet is

$$\mathbf{OPT} = \mathbb{E}_{(\mathbf{y_1}, \dots, \mathbf{y_m}) \sim \mathbf{X_1} \times \dots \times \mathbf{X_m}} \left[\text{OPT}(\mathbf{y_1}, \dots, \mathbf{y_m}) \right].$$

Expected weight of (deterministic) algorithm $\mathcal{A}$ is

$$\mathbf{ALG} = \mathbb{E}_{(\mathbf{y_1}, \dots, \mathbf{y_m}) \sim \mathbf{X_1} \times \dots \times \mathbf{X_m}} \left[\min_\sigma \mathbf{w}(\mathcal{A}(\sigma, \mathbf{y_1}, \dots, \mathbf{y_m})) \right].$$

- With $w(\mathcal{A}(\sigma, y_1, \dots, y_m))$ weight of set outputted by $\mathcal{A}$.
- We assume to have a worst-case arrival order here.

Performance of online algorithm

Performance is measured against that of the prophet.

- Prophet gets to see all realizations $w_i \sim X_i$ after they are sampled.
- Computes (offline) subset S^* with max. weight

$$\text{OPT}(w_1, \dots, w_m) := w(S^*) = max_{S \in \mathcal{F}} \sum_{e \in S} w_e.$$

- Expected weight for prophet is

$$\mathbf{OPT} = \mathbb{E}_{(\mathbf{y_1}, \dots, \mathbf{y_m}) \sim \mathbf{X_1} \times \dots \times \mathbf{X_m}} \left[\text{OPT}(\mathbf{y_1}, \dots, \mathbf{y_m}) \right].$$

Expected weight of (deterministic) algorithm $\mathcal{A}$ is

$$\mathbf{ALG} = \mathbb{E}_{(\mathbf{y_1}, \dots, \mathbf{y_m}) \sim \mathbf{X_1} \times \dots \times \mathbf{X_m}} \left[\min_\sigma \mathbf{w}(\mathcal{A}(\sigma, \mathbf{y_1}, \dots, \mathbf{y_m})) \right].$$

- With $w(\mathcal{A}(\sigma, y_1, \dots, y_m))$ weight of set outputted by $\mathcal{A}$.
- We assume to have a worst-case arrival order here.

For $0 < \alpha < 1$, algorithm $\mathcal{A}$ is α-approximation if

Performance of online algorithm

Performance is measured against that of the prophet.

- Prophet gets to see all realizations $w_i \sim X_i$ after they are sampled.
- Computes (offline) subset S^* with max. weight

$$\text{OPT}(w_1, \ldots, w_m) := w(S^*) = max_{S \in \mathcal{F}} \sum_{e \in S} w_e.$$

- Expected weight for prophet is

$$\mathbf{OPT} = \mathbb{E}_{(\mathbf{y_1}, \ldots, \mathbf{y_m}) \sim \mathbf{X_1} \times \cdots \times \mathbf{X_m}} \left[\text{OPT}(\mathbf{y_1}, \ldots, \mathbf{y_m})\right].$$

Expected weight of (deterministic) algorithm $\mathcal{A}$ is

$$\mathbf{ALG} = \mathbb{E}_{(\mathbf{y_1}, \ldots, \mathbf{y_m}) \sim \mathbf{X_1} \times \cdots \times \mathbf{X_m}} \left[\min_\sigma \mathbf{w}(\mathcal{A}(\sigma, \mathbf{y_1}, \ldots, \mathbf{y_m}))\right].$$

- With $w(\mathcal{A}(\sigma, y_1, \ldots, y_m))$ weight of set outputted by $\mathcal{A}$.
- We assume to have a worst-case arrival order here.

For $0 < \alpha < 1$, algorithm $\mathcal{A}$ is α-approximation if

$$\text{ALG} \geq \alpha \cdot \text{OPT}.$$

Performance of online algorithm

Performance is measured against that of the prophet.

- Prophet gets to see all realizations $w_i \sim X_i$ after they are sampled.
- Computes (offline) subset S^* with max. weight

$$\text{OPT}(w_1, \ldots, w_m) := w(S^*) = max_{S \in \mathcal{F}} \sum_{e \in S} w_e.$$

- Expected weight for prophet is

$$\mathbf{OPT} = \mathbb{E}_{(\mathbf{y_1}, \ldots, \mathbf{y_m}) \sim \mathbf{X_1} \times \cdots \times \mathbf{X_m}} \left[\text{OPT}(\mathbf{y_1}, \ldots, \mathbf{y_m}) \right].$$

Expected weight of (deterministic) algorithm $\mathcal{A}$ is

$$\mathbf{ALG} = \mathbb{E}_{(\mathbf{y_1}, \ldots, \mathbf{y_m}) \sim \mathbf{X_1} \times \cdots \times \mathbf{X_m}} \left[\min_\sigma \mathbf{w}(\mathcal{A}(\sigma, \mathbf{y_1}, \ldots, \mathbf{y_m})) \right].$$

- With $w(\mathcal{A}(\sigma, y_1, \ldots, y_m))$ weight of set outputted by $\mathcal{A}$.
- We assume to have a worst-case arrival order here.

For $0 < \alpha < 1$, algorithm $\mathcal{A}$ is α-approximation if

$$\text{ALG} \geq \alpha \cdot \text{OPT}.$$

This is called a prophet inequality.

Example (cont'd)

$E = \{e_1, e_2\}$ with following distributions.

Example (cont'd)

$E = \{e_1, e_2\}$ with following distributions. Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$.

Example (cont'd)

$E = \{e_1, e_2\}$ with following distributions. Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$. Let

$$w_1 \sim X_1 = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 0 & \text{with probability } 1 - \epsilon \end{cases} \quad (3)$$

Example (cont'd)

$E = \{e_1, e_2\}$ with following distributions. Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$. Let

$$w_1 \sim X_1 = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 0 & \text{with probability } 1 - \epsilon \end{cases} \tag{3}$$

$$w_2 \sim X_2 = \begin{cases} 1 + \delta & \text{with probability } 1. \end{cases} \tag{4}$$

Example (cont'd)

$E = \{e_1, e_2\}$ with following distributions. Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$. Let

$$w_1 \sim X_1 = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 0 & \text{with probability } 1 - \epsilon \end{cases} \quad (3)$$

$$w_2 \sim X_2 = \{ \; 1 + \delta \quad \text{with probability 1.} \quad (4)$$

What can prophet get?

Example (cont'd)

$E = \{e_1, e_2\}$ with following distributions. Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$. Let

$$w_1 \sim X_1 = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 0 & \text{with probability } 1 - \epsilon \end{cases} \tag{3}$$

$$w_2 \sim X_2 = \begin{cases} 1 + \delta & \text{with probability } 1. \end{cases} \tag{4}$$

What can prophet get?

$$\text{OPT}(w_1, w_2) = \max\{w_1, w_2\} = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 1 + \delta & \text{with probability } 1 - \epsilon \end{cases}.$$

Example (cont'd)

$E = \{e_1, e_2\}$ with following distributions. Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$. Let

$$w_1 \sim X_1 = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 0 & \text{with probability } 1 - \epsilon \end{cases} \quad (3)$$

$$w_2 \sim X_2 = \begin{cases} 1 + \delta & \text{with probability } 1. \end{cases} \quad (4)$$

What can prophet get?

$$\text{OPT}(w_1, w_2) = \max\{w_1, w_2\} = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 1 + \delta & \text{with probability } 1 - \epsilon \end{cases}.$$

Then $\mathbb{E}_{\text{OPT}(y_1,y_2)}[\max_i y_i] = \frac{1}{\epsilon} \times \epsilon + (1 + \delta) \times (1 - \epsilon) \to 2 \quad \text{as} \quad \epsilon, \delta \to 0.$

Example (cont'd)

$E = \{e_1, e_2\}$ with following distributions. Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$. Let

$$w_1 \sim X_1 = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 0 & \text{with probability } 1-\epsilon \end{cases} \tag{3}$$

$$w_2 \sim X_2 = \begin{cases} 1+\delta & \text{with probability 1.} \end{cases} \tag{4}$$

What can prophet get?

$$\mathrm{OPT}(w_1, w_2) = \max\{w_1, w_2\} = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 1+\delta & \text{with probability } 1-\epsilon \end{cases}.$$

Then $\mathbb{E}_{\mathrm{OPT}(y_1,y_2)}[\max_i y_i] = \frac{1}{\epsilon} \times \epsilon + (1+\delta) \times (1-\epsilon) \to 2 \quad \text{as} \quad \epsilon, \delta \to 0.$

Optimal algorithm $\mathcal{A}$ is to select e_2 (again, think about it).

Example (cont'd)

$E = \{e_1, e_2\}$ with following distributions. Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$. Let

$$w_1 \sim X_1 = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 0 & \text{with probability } 1 - \epsilon \end{cases} \quad (3)$$

$$w_2 \sim X_2 = \{ 1 + \delta \quad \text{with probability } 1. \quad (4)$$

What can prophet get?

$$\mathrm{OPT}(w_1, w_2) = \max\{w_1, w_2\} = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 1 + \delta & \text{with probability } 1 - \epsilon \end{cases}.$$

Then $\mathbb{E}_{\mathrm{OPT}(y_1, y_2)}[\max_i y_i] = \frac{1}{\epsilon} \times \epsilon + (1 + \delta) \times (1 - \epsilon) \to 2 \quad \text{as} \quad \epsilon, \delta \to 0.$

Optimal algorithm $\mathcal{A}$ is to select e_2 (again, think about it).

- Worst-case order is (e_2, e_1) with $\mathbb{E}_{(y_1, y_2)}[w(\mathcal{A}(\sigma, y_1, y_2))] = 1$.

Example (cont'd)

$E = \{e_1, e_2\}$ with following distributions. Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$. Let

$$w_1 \sim X_1 = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 0 & \text{with probability } 1 - \epsilon \end{cases} \quad (3)$$

$$w_2 \sim X_2 = \{ 1 + \delta \quad \text{with probability } 1. \quad (4)$$

What can prophet get?

$$\mathrm{OPT}(w_1, w_2) = \max\{w_1, w_2\} = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 1 + \delta & \text{with probability } 1 - \epsilon \end{cases}.$$

Then $\mathbb{E}_{\mathrm{OPT}(y_1,y_2)}[\max_i y_i] = \frac{1}{\epsilon} \times \epsilon + (1 + \delta) \times (1 - \epsilon) \to 2$ as $\epsilon, \delta \to 0$.

Optimal algorithm $\mathcal{A}$ is to select e_2 (again, think about it).

- Worst-case order is (e_2, e_1) with $\mathbb{E}_{(y_1,y_2)}[w(\mathcal{A}(\sigma, y_1, y_2))] = 1$.
- I.e., optimal algorithm only half as bad as prophet ($\alpha = \frac{1}{2}$).

Example (cont'd)

$E = \{e_1, e_2\}$ with following distributions. Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$. Let

$$w_1 \sim X_1 = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 0 & \text{with probability } 1 - \epsilon \end{cases} \quad (3)$$

$$w_2 \sim X_2 = \{\ 1 + \delta \quad \text{with probability } 1. \quad (4)$$

What can prophet get?

$$\mathrm{OPT}(w_1, w_2) = \max\{w_1, w_2\} = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 1 + \delta & \text{with probability } 1 - \epsilon \end{cases}.$$

Then $\mathbb{E}_{\mathrm{OPT}(y_1, y_2)}[\max_i y_i] = \frac{1}{\epsilon} \times \epsilon + (1 + \delta) \times (1 - \epsilon) \to 2 \quad \text{as} \quad \epsilon, \delta \to 0$.

Optimal algorithm $\mathcal{A}$ is to select e_2 (again, think about it).

- Worst-case order is (e_2, e_1) with $\mathbb{E}_{(y_1, y_2)}[w(\mathcal{A}(\sigma, y_1, y_2))] = 1$.

- I.e., optimal algorithm only half as bad as prophet ($\alpha = \frac{1}{2}$).
- Also shows that, in general, we cannot hope for $\mathcal{A}$ with $\alpha > \frac{1}{2}$,

Example (cont'd)

$E = \{e_1, e_2\}$ with following distributions. Let $1 > \epsilon, \delta > 0$, and assume that $\frac{1}{\epsilon} > 1 + \delta$. Let

$$w_1 \sim X_1 = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 0 & \text{with probability } 1 - \epsilon \end{cases} \quad (3)$$

$$w_2 \sim X_2 = \{ \; 1 + \delta \quad \text{with probability } 1. \quad (4)$$

What can prophet get?

$$\text{OPT}(w_1, w_2) = \max\{w_1, w_2\} = \begin{cases} \frac{1}{\epsilon} & \text{with probability } \epsilon \\ 1 + \delta & \text{with probability } 1 - \epsilon \end{cases}.$$

Then $\mathbb{E}_{\text{OPT}(y_1,y_2)}[\max_i y_i] = \frac{1}{\epsilon} \times \epsilon + (1 + \delta) \times (1 - \epsilon) \to 2$ as $\epsilon, \delta \to 0$.

Optimal algorithm $\mathcal{A}$ is to select e_2 (again, think about it).

- Worst-case order is (e_2, e_1) with $\mathbb{E}_{(y_1,y_2)}[w(\mathcal{A}(\sigma, y_1, y_2))] = 1$.
- I.e., optimal algorithm only half as bad as prophet ($\alpha = \frac{1}{2}$).
- Also shows that, in general, we cannot hope for $\mathcal{A}$ with $\alpha > \frac{1}{2}$, already in setting where we can select at most one (out of two) elements.

Selecting single element

Prophet Inequality with $\alpha = \frac{1}{2}$

Selecting single item

Krengel, Sucheston and Garling (1978) show there is a prophet inequality with $\alpha = \frac{1}{2}$.

Selecting single item

Krengel, Sucheston and Garling (1978) show there is a prophet inequality with $\alpha = \frac{1}{2}$.

- Simple algorithm was given by Samuel-Cahn (1984):

Selecting single item

Krengel, Sucheston and Garling (1978) show there is a prophet inequality with $\alpha = \frac{1}{2}$.

- Simple algorithm was given by Samuel-Cahn (1984):
  - Set threshold T to be the median of distribution $X_{\max} = \max_i X_i$.

Selecting single item

Krengel, Sucheston and Garling (1978) show there is a prophet inequality with $\alpha = \frac{1}{2}$.

- Simple algorithm was given by Samuel-Cahn (1984):
  - Set threshold T to be the median of distribution $X_{\max} = \max_i X_i$.
  - Select first element e_i whose realized $w_i \sim X_i$ exceeds threshold.

Selecting single item

Krengel, Sucheston and Garling (1978) show there is a prophet inequality with $\alpha = \frac{1}{2}$.

- Simple algorithm was given by Samuel-Cahn (1984):
  - Set threshold T to be the median of distribution $X_{\max} = \max_i X_i$.
  - Select first element e_i whose realized $w_i \sim X_i$ exceeds threshold.

Median of distribution X is value m such that

$$\mathbb{P}(X < m) \leq \frac{1}{2} \quad \text{and} \quad \mathbb{P}(X > m) \leq \frac{1}{2}.$$

Selecting single item

Krengel, Sucheston and Garling (1978) show there is a prophet inequality with $\alpha = \frac{1}{2}$.

- Simple algorithm was given by Samuel-Cahn (1984):
  - Set threshold T to be the median of distribution $X_{\max} = \max_i X_i$.
  - Select first element e_i whose realized $w_i \sim X_i$ exceeds threshold.

Median of distribution X is value m such that

$$\mathbb{P}(X < m) \leq \frac{1}{2} \quad \text{and} \quad \mathbb{P}(X > m) \leq \frac{1}{2}.$$

- For continuous distributions, the median is the "middle value" of the distribution.

Selecting single item

Krengel, Sucheston and Garling (1978) show there is a prophet inequality with $\alpha = \frac{1}{2}$.

- Simple algorithm was given by Samuel-Cahn (1984):
  - Set threshold T to be the median of distribution $X_{\max} = \max_i X_i$.
  - Select first element e_i whose realized $w_i \sim X_i$ exceeds threshold.

Median of distribution X is value m such that

$$\mathbb{P}(X < m) \leq \frac{1}{2} \quad \text{and} \quad \mathbb{P}(X > m) \leq \frac{1}{2}.$$

- For continuous distributions, the median is the "middle value" of the distribution.

Example

Suppose we have uniform distribution over (continuous) interval $[a, b]$. Then $m = \frac{a+b}{2}$.

Kleinberg-Weinberg algorithm

As an alternative to Samuel-Cahn's median-based threshold, Kleinberg and Weinberg (2012) gave another threshold-based algorithm.

Kleinberg-Weinberg algorithm

As an alternative to Samuel-Cahn's median-based threshold, Kleinberg and Weinberg (2012) gave another threshold-based algorithm.

- *Extends to case where multiple elements may be selected under matroid constraint.*

Kleinberg-Weinberg algorithm

As an alternative to Samuel-Cahn's median-based threshold, Kleinberg and Weinberg (2012) gave another threshold-based algorithm.

- *Extends to case where multiple elements may be selected under matroid constraint.*

KW-algorithm for (unknown) arrival order σ

Kleinberg-Weinberg algorithm

As an alternative to Samuel-Cahn's median-based threshold, Kleinberg and Weinberg (2012) gave another threshold-based algorithm.

- *Extends to case where multiple elements may be selected under matroid constraint.*

KW-algorithm for (unknown) arrival order σ

Let X_i be the distribution from which element e_i's weight is drawn.

Kleinberg-Weinberg algorithm

As an alternative to Samuel-Cahn's median-based threshold, Kleinberg and Weinberg (2012) gave another threshold-based algorithm.

- *Extends to case where multiple elements may be selected under matroid constraint.*

KW-algorithm for (unknown) arrival order σ

Let X_i be the distribution from which element e_i's weight is drawn.

- Set threshold

$$T = \frac{\mathbb{E}[\max_j X_j]}{2}.$$

Kleinberg-Weinberg algorithm

As an alternative to Samuel-Cahn's median-based threshold, Kleinberg and Weinberg (2012) gave another threshold-based algorithm.

- *Extends to case where multiple elements may be selected under matroid constraint.*

KW-algorithm for (unknown) arrival order σ

Let X_i be the distribution from which element e_i's weight is drawn.

- Set threshold
$$T = \frac{\mathbb{E}[\max_j X_j]}{2}.$$
- For $i = 1, \ldots, m$: If $w_{\sigma(i)} \geq T$, select $\sigma(i)$ and STOP.

Kleinberg-Weinberg algorithm

As an alternative to Samuel-Cahn's median-based threshold, Kleinberg and Weinberg (2012) gave another threshold-based algorithm.

- *Extends to case where multiple elements may be selected under matroid constraint.*

KW-algorithm for (unknown) arrival order σ

Let X_i be the distribution from which element e_i's weight is drawn.

- Set threshold
$$T = \frac{\mathbb{E}[\max_j X_j]}{2}.$$
- For $i = 1, \ldots, m$: If $w_{\sigma(i)} \geq T$, select $\sigma(i)$ and STOP.

Theorem (Kleinberg and Weinberg, 2012)

The KW-algorithm selects an element e^* with the property that

$$\mathbb{E}_{X_1,\ldots,X_m}[w(e^*)] \geq \frac{1}{2} \cdot \mathbb{E}[\max_j X_j].$$

$$\mathbb{E}_{X_1,\ldots,X_m}[w(e^*)] \geq \frac{\mathbb{E}[\max_j X_j]}{2} \quad (= T)$$

$$\mathbb{E}_{X_1,\ldots,X_m}[w(e^*)] \geq \frac{\mathbb{E}[\max_j X_j]}{2} \ \ (= T)$$

Proof: Let $\tau \in \{1, \ldots, m\}$ be (random) step in which element is select, and let X_τ be the (random) weight of the selected element,

$$\mathbb{E}_{X_1,\ldots,X_m}[w(e^*)] \geq \frac{\mathbb{E}[\max_j X_j]}{2} \ \ (= T)$$

Proof: Let $\tau \in \{1, \ldots, m\}$ be (random) step in which element is select, and let X_τ be the (random) weight of the selected element, i.e., it holds that

$$\mathbb{E}[X_\tau] = \mathbb{E}_{X_1,\ldots,X_m}[w(e^*)]$$

$$\mathbb{E}_{X_1,\ldots,X_m}[w(e^*)] \geq \frac{\mathbb{E}[\max_j X_j]}{2} \;\; (= T)$$

Proof: Let $\tau \in \{1, \ldots, m\}$ be (random) step in which element is select, and let X_τ be the (random) weight of the selected element, i.e., it holds that

$$\mathbb{E}[X_\tau] = \mathbb{E}_{X_1,\ldots,X_m}[w(e^*)]$$

- Assume w.l.o.g. that $\sigma = (e_1, \ldots, e_m)$.

$$\mathbb{E}_{X_1,\ldots,X_m}[w(e^*)] \geq \frac{\mathbb{E}[\max_j X_j]}{2} \quad (= T)$$

Proof: Let $\tau \in \{1, \ldots, m\}$ be (random) step in which element is select, and let X_τ be the (random) weight of the selected element, i.e., it holds that

$$\mathbb{E}[X_\tau] = \mathbb{E}_{X_1,\ldots,X_m}[w(e^*)]$$

- Assume w.l.o.g. that $\sigma = (e_1, \ldots, e_m)$.

It holds that

$$\mathbb{E}[X_\tau] = \int_0^T \mathbb{P}[X_\tau > x]dx + \int_T^\infty \mathbb{P}[X_\tau > x]dx$$

when all distributions X_i are continuous.

$$\mathbb{E}_{X_1,\dots,X_m}[w(e^*)] \geq \frac{\mathbb{E}[\max_j X_j]}{2} \;\; (= T)$$

Proof: Let $\tau \in \{1, \dots, m\}$ be (random) step in which element is select, and let X_τ be the (random) weight of the selected element, i.e., it holds that

$$\mathbb{E}[X_\tau] = \mathbb{E}_{X_1,\dots,X_m}[w(e^*)]$$

- Assume w.l.o.g. that $\sigma = (e_1, \dots, e_m)$.

It holds that

$$\mathbb{E}[X_\tau] = \int_0^T \mathbb{P}[X_\tau > x]dx + \int_T^\infty \mathbb{P}[X_\tau > x]dx$$

when all distributions X_i are continuous.

- See background material for discrete version of this claim:

$$\mathbb{E}_{X_1,\dots,X_m}[w(e^*)] \geq \frac{\mathbb{E}[\max_j X_j]}{2} \quad (= T)$$

Proof: Let $\tau \in \{1,\dots,m\}$ be (random) step in which element is select, and let X_τ be the (random) weight of the selected element, i.e., it holds that

$$\mathbb{E}[X_\tau] = \mathbb{E}_{X_1,\dots,X_m}[w(e^*)]$$

- Assume w.l.o.g. that $\sigma = (e_1,\dots,e_m)$.

It holds that

$$\mathbb{E}[X_\tau] = \int_0^T \mathbb{P}[X_\tau > x]dx + \int_T^\infty \mathbb{P}[X_\tau > x]dx$$

when all distributions X_i are continuous.

- See background material for discrete version of this claim:

$$\mathbb{E}[X] = \sum_{k=0}^{\infty} \mathbb{P}[X \geq k].$$

$$\mathbb{E}[X_\tau] = \int_0^T \mathbb{P}[X_\tau > x]dx + \int_T^\infty \mathbb{P}[X_\tau > x]dx, \quad T = \frac{\mathbb{E}[\max_j X_j]}{2}$$

$$\mathbb{E}[X_\tau] = \int_0^T \mathbb{P}[X_\tau > x]dx + \int_T^\infty \mathbb{P}[X_\tau > x]dx, \quad T = \frac{\mathbb{E}[\max_j X_j]}{2}$$

Let $p = \mathbb{P}[\max_j X_j \geq T]$.

$$\mathbb{E}[X_\tau] = \int_0^T \mathbb{P}[X_\tau > x]dx + \int_T^\infty \mathbb{P}[X_\tau > x]dx, \quad T = \frac{\mathbb{E}[\max_j X_j]}{2}$$

Let $p = \mathbb{P}[\max_j X_j \geq T]$.

- $1 - p$ is probability that we do not select anything.

$$\mathbb{E}[X_\tau] = \int_0^T \mathbb{P}[X_\tau > x]dx + \int_T^\infty \mathbb{P}[X_\tau > x]dx, \quad T = \frac{\mathbb{E}[\max_j X_j]}{2}$$

Let $p = \mathbb{P}[\max_j X_j \geq T]$.

- $1 - p$ is probability that we do not select anything.
- For any $i = 1, \ldots, m$, probability that we have not selected an element in step i is then at least $1 - p$.

$$\mathbb{E}[X_\tau] = \int_0^T \mathbb{P}[X_\tau > x]dx + \int_T^\infty \mathbb{P}[X_\tau > x]dx, \qquad T = \frac{\mathbb{E}[\max_j X_j]}{2}$$

Let $p = \mathbb{P}[\max_j X_j \geq T]$.

- $1 - p$ is probability that we do not select anything.
- For any $i = 1, \ldots, m$, probability that we have not selected an element in step i is then at least $1 - p$.

It is not hard to see that

$$\int_0^T \mathbb{P}[X_\tau > x]dx \geq \int_0^T \mathbb{P}[X_\tau > T]dx \geq \int_0^T p \cdot dx = pT. \tag{5}$$

$$\mathbb{E}[X_\tau] = \int_0^T \mathbb{P}[X_\tau > x]dx + \int_T^\infty \mathbb{P}[X_\tau > x]dx, \quad T = \frac{\mathbb{E}[\max_j X_j]}{2}$$

Let $p = \mathbb{P}[\max_j X_j \geq T]$.

- $1 - p$ is probability that we do not select anything.
- For any $i = 1, \ldots, m$, probability that we have not selected an element in step i is then at least $1 - p$.

It is not hard to see that

$$\int_0^T \mathbb{P}[X_\tau > x]dx \geq \int_0^T \mathbb{P}[X_\tau > T]dx \geq \int_0^T p \cdot dx = pT. \tag{5}$$

Furthermore, for $x \geq T$ it holds that

$$\mathbb{E}[X_\tau] = \int_0^T \mathbb{P}[X_\tau > x]dx + \int_T^\infty \mathbb{P}[X_\tau > x]dx, \qquad T = \frac{\mathbb{E}[\max_j X_j]}{2}$$

Let $p = \mathbb{P}[\max_j X_j \geq T]$.

- $1 - p$ is probability that we do not select anything.
- For any $i = 1, \ldots, m$, probability that we have not selected an element in step i is then at least $1 - p$.

It is not hard to see that

$$\int_0^T \mathbb{P}[X_\tau > x]dx \geq \int_0^T \mathbb{P}[X_\tau > T]dx \geq \int_0^T p \cdot dx = pT. \tag{5}$$

Furthermore, for $x \geq T$ it holds that

$$\mathbb{P}[X_\tau > x] \;\; = \;\; \sum_{j=1}^{m} \mathbb{P}[X_\tau > x \mid \tau = j]\mathbb{P}[\tau = j]$$

$$\mathbb{E}[X_\tau] = \int_0^T \mathbb{P}[X_\tau > x]dx + \int_T^\infty \mathbb{P}[X_\tau > x]dx, \qquad T = \frac{\mathbb{E}[\max_j X_j]}{2}$$

Let $p = \mathbb{P}[\max_j X_j \geq T]$.

- $1 - p$ is probability that we do not select anything.
- For any $i = 1, \ldots, m$, probability that we have not selected an element in step i is then at least $1 - p$.

It is not hard to see that

$$\int_0^T \mathbb{P}[X_\tau > x]dx \geq \int_0^T \mathbb{P}[X_\tau > T]dx \geq \int_0^T p \cdot dx = pT. \tag{5}$$

Furthermore, for $x \geq T$ it holds that

$$\begin{aligned} \mathbb{P}[X_\tau > x] &= \sum_{j=1}^{m} \mathbb{P}[X_\tau > x \mid \tau = j]\mathbb{P}[\tau = j] \\ &\geq (1-p)\sum_{j=1}^{m} \mathbb{P}[X_j > x] \end{aligned}$$

$$\mathbb{E}[X_\tau] = \int_0^T \mathbb{P}[X_\tau > x]dx + \int_T^\infty \mathbb{P}[X_\tau > x]dx, \quad T = \frac{\mathbb{E}[\max_j X_j]}{2}$$

Let $p = \mathbb{P}[\max_j X_j \geq T]$.

- $1 - p$ is probability that we do not select anything.
- For any $i = 1, \ldots, m$, probability that we have not selected an element in step i is then at least $1 - p$.

It is not hard to see that

$$\int_0^T \mathbb{P}[X_\tau > x]dx \geq \int_0^T \mathbb{P}[X_\tau > T]dx \geq \int_0^T p \cdot dx = pT. \tag{5}$$

Furthermore, for $x \geq T$ it holds that

$$\begin{aligned} \mathbb{P}[X_\tau > x] &= \sum_{j=1}^m \mathbb{P}[X_\tau > x \mid \tau = j]\mathbb{P}[\tau = j] \\ &\geq (1-p)\sum_{j=1}^m \mathbb{P}[X_j > x] \\ &\geq (1-p)\mathbb{P}[\max_j X_j > x] \qquad \text{(union bound)} \end{aligned}$$

$$\mathbb{E}[X_\tau] \geq pT + (1-p)\int_T^\infty \mathbb{P}[\max_j X_j > x]dx, \quad T = \frac{\mathbb{E}[\max_j X_j]}{2}$$

$$\mathbb{E}[X_\tau] \geq pT + (1-p)\int_T^\infty \mathbb{P}[\max_j X_j > x]dx, \quad T = \frac{\mathbb{E}[\max_j X_j]}{2}$$

Note that

$$\mathbb{E}[\max_j X_j] = \int_0^T \mathbb{P}[\max_j X_j > x]dx + \int_T^\infty \mathbb{P}[\max_j X_j > x]dx = 2T$$

by definition of T.

$$\mathbb{E}[X_\tau] \geq pT + (1-p)\int_T^\infty \mathbb{P}[\max_j X_j > x]dx, \quad T = \frac{\mathbb{E}[\max_j X_j]}{2}$$

Note that

$$\mathbb{E}[\max_j X_j] = \int_0^T \mathbb{P}[\max_j X_j > x]dx + \int_T^\infty \mathbb{P}[\max_j X_j > x]dx = 2T$$

by definition of T. Since $\int_0^T \mathbb{P}[\max_j X_j > x]dx \leq T$,

$$\mathbb{E}[X_\tau] \geq pT + (1-p)\int_T^\infty \mathbb{P}[\max_j X_j > x]dx, \quad T = \frac{\mathbb{E}[\max_j X_j]}{2}$$

Note that

$$\mathbb{E}[\max_j X_j] = \int_0^T \mathbb{P}[\max_j X_j > x]dx + \int_T^\infty \mathbb{P}[\max_j X_j > x]dx = 2T$$

by definition of T. Since $\int_0^T \mathbb{P}[\max_j X_j > x]dx \leq T$, it holds that

$$\int_T^\infty \mathbb{P}[\max_j X_j > x]dx \geq T.$$

$$\mathbb{E}[X_\tau] \geq pT + (1-p)\int_T^\infty \mathbb{P}[\max_j X_j > x]dx, \quad T = \frac{\mathbb{E}[\max_j X_j]}{2}$$

Note that

$$\mathbb{E}[\max_j X_j] = \int_0^T \mathbb{P}[\max_j X_j > x]dx + \int_T^\infty \mathbb{P}[\max_j X_j > x]dx = 2T$$

by definition of T. Since $\int_0^T \mathbb{P}[\max_j X_j > x]dx \leq T$, it holds that

$$\int_T^\infty \mathbb{P}[\max_j X_j > x]dx \geq T.$$

Plugging this into the main inequality above gives

$$\mathbb{E}[X_\tau] \geq pT + (1-p)T = T = \frac{\mathbb{E}[\max_j X_j]}{2}.$$

$$\mathbb{E}[X_\tau] \geq pT + (1-p)\int_T^\infty \mathbb{P}[\max_j X_j > x]dx, \quad T = \frac{\mathbb{E}[\max_j X_j]}{2}$$

Note that

$$\mathbb{E}[\max_j X_j] = \int_0^T \mathbb{P}[\max_j X_j > x]dx + \int_T^\infty \mathbb{P}[\max_j X_j > x]dx = 2T$$

by definition of T. Since $\int_0^T \mathbb{P}[\max_j X_j > x]dx \leq T$, it holds that

$$\int_T^\infty \mathbb{P}[\max_j X_j > x]dx \geq T.$$

Plugging this into the main inequality above gives

$$\mathbb{E}[X_\tau] \geq pT + (1-p)T = T = \frac{\mathbb{E}[\max_j X_j]}{2}.$$

This completes the proof. □

Some remarks

Theorem (Kleinberg and Weinberg, 2012)

The KW-algorithm selects an element e^* with the property that

$$\mathbb{E}_{X_1,\dots,X_m}[w(e^*)] \geq \frac{1}{2} \cdot \mathbb{E}[\max_j X_j].$$

Some remarks

Theorem (Kleinberg and Weinberg, 2012)

The KW-algorithm selects an element e^* with the property that

$$\mathbb{E}_{X_1,\dots,X_m}[w(e^*)] \geq \frac{1}{2} \cdot \mathbb{E}[\max_j X_j].$$

- Algorithm is optimal trade-off between weight of selected elements and probability of selecting an element.

Some remarks

Theorem (Kleinberg and Weinberg, 2012)

The KW-algorithm selects an element e^* with the property that

$$\mathbb{E}_{X_1,\dots,X_m}[w(e^*)] \geq \frac{1}{2} \cdot \mathbb{E}[\max_j X_j].$$

- Algorithm is optimal trade-off between weight of selected elements and probability of selecting an element.
  - Higher threshold would give better weight of selected element, but prob. that we can select one gets smaller.

Some remarks

Theorem (Kleinberg and Weinberg, 2012)

The KW-algorithm selects an element e^* with the property that

$$\mathbb{E}_{X_1,\ldots,X_m}[w(e^*)] \geq \frac{1}{2} \cdot \mathbb{E}[\max_j X_j].$$

- Algorithm is optimal trade-off between weight of selected elements and probability of selecting an element.
  - Higher threshold would give better weight of selected element, but prob. that we can select one gets smaller.
  - Lower threshold would increase prob. of selecting element, but weight will be lower.

Some remarks

Theorem (Kleinberg and Weinberg, 2012)

The KW-algorithm selects an element e^* with the property that

$$\mathbb{E}_{X_1,\dots,X_m}[w(e^*)] \geq \frac{1}{2} \cdot \mathbb{E}[\max_j X_j].$$

- Algorithm is optimal trade-off between weight of selected elements and probability of selecting an element.
  - Higher threshold would give better weight of selected element, but prob. that we can select one gets smaller.
  - Lower threshold would increase prob. of selecting element, but weight will be lower.
- Yields strategy proof online mechanism (in appropriate model).

Some remarks

Theorem (Kleinberg and Weinberg, 2012)

The KW-algorithm selects an element e^* with the property that

$$\mathbb{E}_{X_1,\dots,X_m}[w(e^*)] \geq \frac{1}{2} \cdot \mathbb{E}[\max_j X_j].$$

- Algorithm is optimal trade-off between weight of selected elements and probability of selecting an element.
  - Higher threshold would give better weight of selected element, but prob. that we can select one gets smaller.
  - Lower threshold would increase prob. of selecting element, but weight will be lower.
- Yields strategy proof online mechanism (in appropriate model).
  - Give item to first bidder exceeding threshold, and charge price T.

Some remarks

Theorem (Kleinberg and Weinberg, 2012)

The KW-algorithm selects an element e^* with the property that

$$\mathbb{E}_{X_1,\ldots,X_m}[w(e^*)] \geq \frac{1}{2} \cdot \mathbb{E}[\max_j X_j].$$

- Algorithm is optimal trade-off between weight of selected elements and probability of selecting an element.
  - Higher threshold would give better weight of selected element, but prob. that we can select one gets smaller.
  - Lower threshold would increase prob. of selecting element, but weight will be lower.
- Yields strategy proof online mechanism (in appropriate model).
  - Give item to first bidder exceeding threshold, and charge price T.
  - Similar to what we saw for secretary problem.

Matroid prophet inequality

Matroid prophet inequality

Selecting indep. set from matroid $\mathcal{M} = (E, \mathcal{I})$ with arrival order σ.

Set $S = \emptyset$.

- For $i = 1, \dots, m$, a realization $w_i \sim X_i$ is generated.
  - All realizations w_i are shown to the adversary.
- For $i = 1, \dots, m$:
  - Adversary chooses $\sigma(i) \in E$, and reveals it and its weight w_i.
  - Online algorithm $\mathcal{A}$ decides whether to accept or reject $\sigma(i)$, where acceptance is only allowed if $S + \sigma(i) \in \mathcal{I}$.

Matroid prophet inequality

Selecting indep. set from matroid $\mathcal{M} = (E, \mathcal{I})$ with arrival order σ.

Set $S = \emptyset$.

- For $i = 1, \ldots, m$, a realization $w_i \sim X_i$ is generated.
  - All realizations w_i are shown to the adversary.
- For $i = 1, \ldots, m$:
  - Adversary chooses $\sigma(i) \in E$, and reveals it and its weight w_i.
  - Online algorithm $\mathcal{A}$ decides whether to accept or reject $\sigma(i)$, where acceptance is only allowed if $S + \sigma(i) \in \mathcal{I}$.

Theorem (Kleinberg-Weinberg, 2012)

There is an online algorithm $\mathcal{A}$ for selecting multiple elements subject to a matroid constraint (under adversarial arrival order), with

$$ALG(\mathcal{A}) \geq \frac{1}{2} \cdot OPT,$$

Matroid prophet inequality

Selecting indep. set from matroid $\mathcal{M} = (E, \mathcal{I})$ with arrival order σ.

Set $S = \emptyset$.

- For $i = 1, \ldots, m$, a realization $w_i \sim X_i$ is generated.
  - All realizations w_i are shown to the adversary.
- For $i = 1, \ldots, m$:
  - Adversary chooses $\sigma(i) \in E$, and reveals it and its weight w_i.
  - Online algorithm $\mathcal{A}$ decides whether to accept or reject $\sigma(i)$, where acceptance is only allowed if $S + \sigma(i) \in \mathcal{I}$.

Theorem (Kleinberg-Weinberg, 2012)

There is an online algorithm $\mathcal{A}$ for selecting multiple elements subject to a matroid constraint (under adversarial arrival order), with

$$ALG(\mathcal{A}) \geq \frac{1}{2} \cdot OPT,$$

where $OPT = \mathbb{E}_{(y_1,\ldots,y_m)\sim X_1\times\cdots\times X_m}[OPT(y_1, \ldots, y_m)]$ is offline optimum.

KW-algorithm for matroid constraint

Algorithm sets threshold in step i based on marginal contribution of $\sigma(i)$.

KW-algorithm for matroid constraint

Algorithm sets threshold in step i based on marginal contribution of $\sigma(i)$.

- Let $y' = (y'_1, \ldots, y'_m) \geq 0$ be given weights, and let B' be a max. weight base under y'.

KW-algorithm for matroid constraint

Algorithm sets threshold in step i based on marginal contribution of $\sigma(i)$.

- Let $y' = (y'_1, \ldots, y'_m) \geq 0$ be given weights, and let B' be a max. weight base under y'.
- For given independent set $S \in \mathcal{I}$, we can augment S with elements $R(S) \subseteq B'$ so that $S \cup R(S)$ is base of $\mathcal{M}$.

KW-algorithm for matroid constraint

Algorithm sets threshold in step i based on marginal contribution of $\sigma(i)$.

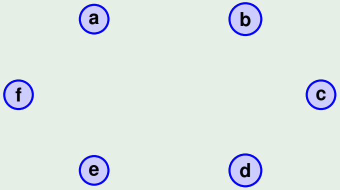
- Let $y' = (y'_1, \ldots, y'_m) \geq 0$ be given weights, and let B' be a max. weight base under y'.
- For given independent set $S \in \mathcal{I}$, we can augment S with elements $R(S) \subseteq B'$ so that $S \cup R(S)$ is base of $\mathcal{M}$.
  - Choose R so that $y'(R)$ is maximized (among all choices for R).

KW-algorithm for matroid constraint

Algorithm sets threshold in step i based on marginal contribution of $\sigma(i)$.

- Let $y' = (y'_1, \ldots, y'_m) \geq 0$ be given weights, and let B' be a max. weight base under y'.
- For given independent set $S \in \mathcal{I}$, we can augment S with elements $R(S) \subseteq B'$ so that $S \cup R(S)$ is base of $\mathcal{M}$.
  - Choose R so that $y'(R)$ is maximized (among all choices for R).

Example (Graphic matroid)

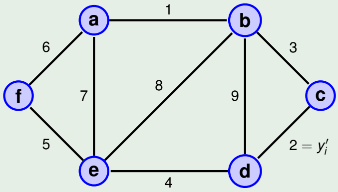


KW-algorithm for matroid constraint

Algorithm sets threshold in step i based on marginal contribution of $\sigma(i)$.

- Let $y' = (y'_1, \ldots, y'_m) \geq 0$ be given weights, and let B' be a max. weight base under y'.
- For given independent set $S \in \mathcal{I}$, we can augment S with elements $R(S) \subseteq B'$ so that $S \cup R(S)$ is base of $\mathcal{M}$.
  - Choose R so that $y'(R)$ is maximized (among all choices for R).

Example (Graphic matroid)

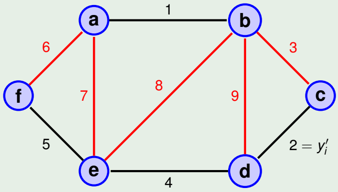


KW-algorithm for matroid constraint

Algorithm sets threshold in step i based on marginal contribution of $\sigma(i)$.

- Let $y' = (y'_1, \ldots, y'_m) \geq 0$ be given weights, and let B' be a max. weight base under y'.
- For given independent set $S \in \mathcal{I}$, we can augment S with elements $R(S) \subseteq B'$ so that $S \cup R(S)$ is base of $\mathcal{M}$.
  - Choose R so that $y'(R)$ is maximized (among all choices for R).

Example (Graphic matroid)



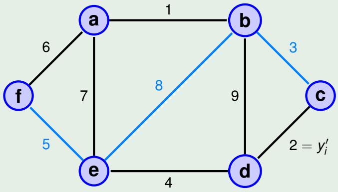
Base B'

KW-algorithm for matroid constraint

Algorithm sets threshold in step i based on marginal contribution of $\sigma(i)$.

- Let $y' = (y'_1, \ldots, y'_m) \geq 0$ be given weights, and let B' be a max. weight base under y'.
- For given independent set $S \in \mathcal{I}$, we can augment S with elements $R(S) \subseteq B'$ so that $S \cup R(S)$ is base of $\mathcal{M}$.
  - Choose R so that $y'(R)$ is maximized (among all choices for R).

Example (Graphic matroid)



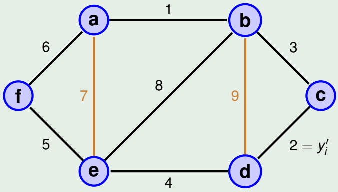
S selected so far

KW-algorithm for matroid constraint

Algorithm sets threshold in step i based on marginal contribution of $\sigma(i)$.

- Let $y' = (y'_1, \ldots, y'_m) \geq 0$ be given weights, and let B' be a max. weight base under y'.
- For given independent set $S \in \mathcal{I}$, we can augment S with elements $R(S) \subseteq B'$ so that $S \cup R(S)$ is base of $\mathcal{M}$.
  - Choose R so that $y'(R)$ is maximized (among all choices for R).

Example (Graphic matroid)



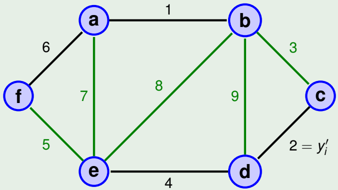
Augmentation $R(S)$

KW-algorithm for matroid constraint

Algorithm sets threshold in step i based on marginal contribution of $\sigma(i)$.

- Let $y' = (y'_1, \dots, y'_m) \geq 0$ be given weights, and let B' be a max. weight base under y'.
- For given independent set $S \in \mathcal{I}$, we can augment S with elements $R(S) \subseteq B'$ so that $S \cup R(S)$ is base of $\mathcal{M}$.
  - Choose R so that $y'(R)$ is maximized (among all choices for R).

Example (Graphic matroid)



$R(S) \cup S$

Assume that $\sigma = (e_1, \ldots, e_m)$.

Assume that $\sigma = (e_1, \ldots, e_m)$.

KW-algorithm with initial $S = \emptyset$

For $i = 1, \ldots, m$: If $S \cup \{e_i\} \in \mathcal{I}$ do the following.

- Set threshold
$$T_i = \mathbb{E}_{y' \sim X_1 \times \cdots \times X_m}[y'(R(S)) - y'(R(S \cup \{e_i\}))].$$
- Set $S \leftarrow S \cup \{e_i\}$ if $w_i \geq T_i$.

Assume that $\sigma = (e_1, \ldots, e_m)$.

KW-algorithm with initial $S = \emptyset$

For $i = 1, \ldots, m$: If $S \cup \{e_i\} \in \mathcal{I}$ do the following.

- Set threshold
$$T_i = \mathbb{E}_{y' \sim X_1 \times \cdots \times X_m}[y'(R(S)) - y'(R(S \cup \{e_i\}))].$$
- Set $S \leftarrow S \cup \{e_i\}$ if $w_i \geq T_i$.

Roughly speaking, T_i is expected gain of adding e_i to S.

Assume that $\sigma = (e_1, \ldots, e_m)$.

KW-algorithm with initial $S = \emptyset$

For $i = 1, \ldots, m$: If $S \cup \{e_i\} \in \mathcal{I}$ do the following.

- Set threshold
$$T_i = \mathbb{E}_{y' \sim X_1 \times \cdots \times X_m}[y'(R(S)) - y'(R(S \cup \{e_i\}))].$$
- Set $S \leftarrow S \cup \{e_i\}$ if $w_i \geq T_i$.

Roughly speaking, T_i is expected gain of adding e_i to S.

- If revealed realization w_i exceeds expected gain, add it to S.

Assume that $\sigma = (e_1, \ldots, e_m)$.

KW-algorithm with initial $S = \emptyset$

For $i = 1, \ldots, m$: If $S \cup \{e_i\} \in \mathcal{I}$ do the following.

- Set threshold
$$T_i = \mathbb{E}_{y' \sim X_1 \times \cdots \times X_m}[y'(R(S)) - y'(R(S \cup \{e_i\}))].$$
- Set $S \leftarrow S \cup \{e_i\}$ if $w_i \geq T_i$.

Roughly speaking, T_i is expected gain of adding e_i to S.

- If revealed realization w_i exceeds expected gain, add it to S.

In order to determine T_i, we take expectation over all elements (and not just those that have not yet arrived).

Assume that $\sigma = (e_1, \ldots, e_m)$.

KW-algorithm with initial $S = \emptyset$

For $i = 1, \ldots, m$: If $S \cup \{e_i\} \in \mathcal{I}$ do the following.

- Set threshold
$$T_i = \mathbb{E}_{y' \sim X_1 \times \cdots \times X_m}[y'(R(S)) - y'(R(S \cup \{e_i\}))].$$
- Set $S \leftarrow S \cup \{e_i\}$ if $w_i \geq T_i$.

Roughly speaking, T_i is expected gain of adding e_i to S.

- If revealed realization w_i exceeds expected gain, add it to S.

In order to determine T_i, we take expectation over all elements (and not just those that have not yet arrived).

- T_i does not use realized weights $w_1, \ldots, w_{i-1}$ revealed so far.

Assume that $\sigma = (e_1, \ldots, e_m)$.

KW-algorithm with initial $S = \emptyset$

For $i = 1, \ldots, m$: If $S \cup \{e_i\} \in \mathcal{I}$ do the following.

- Set threshold
$$T_i = \mathbb{E}_{y' \sim X_1 \times \cdots \times X_m}[y'(R(S)) - y'(R(S \cup \{e_i\}))].$$
- Set $S \leftarrow S \cup \{e_i\}$ if $w_i \geq T_i$.

Roughly speaking, T_i is expected gain of adding e_i to S.

- If revealed realization w_i exceeds expected gain, add it to S.

In order to determine T_i, we take expectation over all elements (and not just those that have not yet arrived).

- T_i does not use realized weights $w_1, \ldots, w_{i-1}$ revealed so far.

Computational remark: *If the X_i are discrete (with finite support), T_i can be computed exactly (in possibly exponential time).*

Assume that $\sigma = (e_1, \dots, e_m)$.

KW-algorithm with initial $S = \emptyset$

For $i = 1, \dots, m$: If $S \cup \{e_i\} \in \mathcal{I}$ do the following.

- Set threshold

$$T_i = \mathbb{E}_{y' \sim X_1 \times \dots \times X_m}[y'(R(S)) - y'(R(S \cup \{e_i\}))].$$

- Set $S \leftarrow S \cup \{e_i\}$ if $w_i \geq T_i$.

Roughly speaking, T_i is expected gain of adding e_i to S.

- If revealed realization w_i exceeds expected gain, add it to S.

In order to determine T_i, we take expectation over all elements (and not just those that have not yet arrived).

- T_i does not use realized weights $w_1, \dots, w_{i-1}$ revealed so far.

Computational remark: *If the X_i are discrete (with finite support), T_i can be computed exactly (in possibly exponential time). For continuous distributions, usually approximation is needed*

Assume that $\sigma = (e_1, \dots, e_m)$.

KW-algorithm with initial $S = \emptyset$

For $i = 1, \dots, m$: If $S \cup \{e_i\} \in \mathcal{I}$ do the following.

- Set threshold
$$T_i = \mathbb{E}_{y' \sim X_1 \times \cdots \times X_m}[y'(R(S)) - y'(R(S \cup \{e_i\}))].$$
- Set $S \leftarrow S \cup \{e_i\}$ if $w_i \geq T_i$.

Roughly speaking, T_i is expected gain of adding e_i to S.

- If revealed realization w_i exceeds expected gain, add it to S.

In order to determine T_i, we take expectation over all elements (and not just those that have not yet arrived).

- T_i does not use realized weights $w_1, \dots, w_{i-1}$ revealed so far.

Computational remark: *If the X_i are discrete (with finite support), T_i can be computed exactly (in possibly exponential time). For continuous distributions, usually approximation is needed (by means of repeatedly sampling vectors y' from $\times_i X_i$ and computing average).*

Remarks

Theorem (Kleinberg and Weinberg, 2012)

KW-algorithm for matroids gives prophet inequality with $\alpha = \frac{1}{2}$.

Remarks

Theorem (Kleinberg and Weinberg, 2012)

KW-algorithm for matroids gives prophet inequality with $\alpha = \frac{1}{2}$.

- Result also extends to intersection of p matroid constraints, where one then gets $\alpha = 1/(4p - 2)$.

Remarks

Theorem (Kleinberg and Weinberg, 2012)

KW-algorithm for matroids gives prophet inequality with $\alpha = \frac{1}{2}$.

- Result also extends to intersection of p matroid constraints, where one then gets $\alpha = 1/(4p - 2)$.
- Can be used to model, e.g., setting where edges of bipartite graph arrive online (with known distributions).

Remarks

Theorem (Kleinberg and Weinberg, 2012)

KW-algorithm for matroids gives prophet inequality with $\alpha = \frac{1}{2}$.

- Result also extends to intersection of p matroid constraints, where one then gets $\alpha = 1/(4p - 2)$.
- Can be used to model, e.g., setting where edges of bipartite graph arrive online (with known distributions).

Strategyproof mechanism?

Remarks

Theorem (Kleinberg and Weinberg, 2012)

KW-algorithm for matroids gives prophet inequality with $\alpha = \frac{1}{2}$.

- Result also extends to intersection of p matroid constraints, where one then gets $\alpha = 1/(4p - 2)$.
- Can be used to model, e.g., setting where edges of bipartite graph arrive online (with known distributions).

Strategyproof mechanism?

- For single element setting, conversion of respective KW-algorithm into strategyproof mechanism is easy.

Remarks

Theorem (Kleinberg and Weinberg, 2012)

KW-algorithm for matroids gives prophet inequality with $\alpha = \frac{1}{2}$.

- Result also extends to intersection of p matroid constraints, where one then gets $\alpha = 1/(4p - 2)$.
- Can be used to model, e.g., setting where edges of bipartite graph arrive online (with known distributions).

Strategyproof mechanism?

- For single element setting, conversion of respective KW-algorithm into strategyproof mechanism is easy.
- This is not the case for the matroid setting.

Remarks

Theorem (Kleinberg and Weinberg, 2012)

KW-algorithm for matroids gives prophet inequality with $\alpha = \frac{1}{2}$.

- Result also extends to intersection of p matroid constraints, where one then gets $\alpha = 1/(4p - 2)$.
- Can be used to model, e.g., setting where edges of bipartite graph arrive online (with known distributions).

Strategyproof mechanism?

- For single element setting, conversion of respective KW-algorithm into strategyproof mechanism is easy.
- This is not the case for the matroid setting.

Adaptive vs. non-adaptive threshold-based algorithms.

Remarks

Theorem (Kleinberg and Weinberg, 2012)

KW-algorithm for matroids gives prophet inequality with $\alpha = \frac{1}{2}$.

- Result also extends to intersection of p matroid constraints, where one then gets $\alpha = 1/(4p - 2)$.
- Can be used to model, e.g., setting where edges of bipartite graph arrive online (with known distributions).

Strategyproof mechanism?

- For single element setting, conversion of respective KW-algorithm into strategyproof mechanism is easy.
- This is not the case for the matroid setting.

Adaptive vs. non-adaptive threshold-based algorithms.

- KW-algorithm is adaptive in the sense that threshold T_i in step i depends on arrival order σ and elements S selected so far.

Remarks

Theorem (Kleinberg and Weinberg, 2012)

KW-algorithm for matroids gives prophet inequality with $\alpha = \frac{1}{2}$.

- Result also extends to intersection of p matroid constraints, where one then gets $\alpha = 1/(4p - 2)$.
- Can be used to model, e.g., setting where edges of bipartite graph arrive online (with known distributions).

Strategyproof mechanism?

- For single element setting, conversion of respective KW-algorithm into strategyproof mechanism is easy.
- This is not the case for the matroid setting.

Adaptive vs. non-adaptive threshold-based algorithms.

- KW-algorithm is adaptive in the sense that threshold T_i in step i depends on arrival order σ and elements S selected so far.
  - Does not necessarily yield strategyproof (online) mechanism.

Non-adaptive threshold-based algorithms

A non-adaptive threshold-based algorithm sets threshold $T(e)$ for every $e \in E$ before start of the algorithm (independent of i).

Non-adaptive threshold-based algorithms

A non-adaptive threshold-based algorithm sets threshold $T(e)$ for every $e \in E$ before start of the algorithm (independent of i).

- It then selects *every* element whose weight exceeds the threshold (and that preserves independence).

Non-adaptive threshold-based algorithms

A non-adaptive threshold-based algorithm sets threshold $T(e)$ for every $e \in E$ before start of the algorithm (independent of i).

- It then selects *every* element whose weight exceeds the threshold (and that preserves independence).

Gives rise to so-called order-oblivious posted price mechanisms.

Non-adaptive threshold-based algorithms

A non-adaptive threshold-based algorithm sets threshold $T(e)$ for every $e \in E$ before start of the algorithm (independent of i).

- It then selects *every* element whose weight exceeds the threshold (and that preserves independence).

Gives rise to so-called order-oblivious posted price mechanisms.

- See Chawla, Goldner, Karlin and Miller (2020) for a (recent) algorithm for graphic matroids.

Non-adaptive threshold-based algorithms

A non-adaptive threshold-based algorithm sets threshold $T(e)$ for every $e \in E$ before start of the algorithm (independent of i).

- It then selects *every* element whose weight exceeds the threshold (and that preserves independence).

Gives rise to so-called order-oblivious posted price mechanisms.

- See Chawla, Goldner, Karlin and Miller (2020) for a (recent) algorithm for graphic matroids.

Interestingly, there exist matroid constraints for which *no* non-adaptive threshold-based algorithm can exist.

Non-adaptive threshold-based algorithms

A non-adaptive threshold-based algorithm sets threshold $T(e)$ for every $e \in E$ before start of the algorithm (independent of i).

- It then selects *every* element whose weight exceeds the threshold (and that preserves independence).

Gives rise to so-called order-oblivious posted price mechanisms.

- See Chawla, Goldner, Karlin and Miller (2020) for a (recent) algorithm for graphic matroids.

Interestingly, there exist matroid constraints for which *no* non-adaptive threshold-based algorithm can exist.

- Feldman, Svensson and Zenklusen (2020) give such an example for so-called gammoids.

Non-adaptive threshold-based algorithms

A non-adaptive threshold-based algorithm sets threshold $T(e)$ for every $e \in E$ before start of the algorithm (independent of i).

- It then selects *every* element whose weight exceeds the threshold (and that preserves independence).

Gives rise to so-called order-oblivious posted price mechanisms.

- See Chawla, Goldner, Karlin and Miller (2020) for a (recent) algorithm for graphic matroids.

Interestingly, there exist matroid constraints for which *no* non-adaptive threshold-based algorithm can exist.

- Feldman, Svensson and Zenklusen (2020) give such an example for so-called gammoids.
- They show that one can hope at best for a prophet inequality with

$$\alpha = \Omega\left(\frac{\log\log(m)}{\log(m)}\right).$$

Beyond matroids

Beyond matroids

For general downward-closed set systems, lower bound from last week also applies to Bayesian setting (with adversarial arrivals).

Beyond matroids

For general downward-closed set systems, lower bound from last week also applies to Bayesian setting (with adversarial arrivals).

Theorem (Babaioff et al. (2007), Rubinstein (2016))

There is no randomized algorithm that, for every downward-closed set system $\mathcal{F} = (E, \mathcal{I})$ with m elements having known weight distribution, obtains a prophet inequality with α better than

$$\alpha = \Omega\left(\frac{\log\log(m)}{\log(m)}\right)$$

Selecting single element

Sample-based threshold

What prior information is needed?

What prior information is needed?

Remember that the KW-algorithm for selecting a single item uses the threshold

$$T = \frac{\mathbb{E}[\max_j X_j]}{2}.$$

What prior information is needed?

Remember that the KW-algorithm for selecting a single item uses the threshold

$$T = \frac{\mathbb{E}[\max_j X_j]}{2}.$$

Computing threshold requires full knowledge of the distributions X_i.

What prior information is needed?

Remember that the KW-algorithm for selecting a single item uses the threshold

$$T = \frac{\mathbb{E}[\max_j X_j]}{2}.$$

Computing threshold requires full knowledge of the distributions X_i.

- Can be non-trivial depending on what the distributions look like.

What prior information is needed?

Remember that the KW-algorithm for selecting a single item uses the threshold

$$T = \frac{\mathbb{E}[\max_j X_j]}{2}.$$

Computing threshold requires full knowledge of the distributions X_i.

- Can be non-trivial depending on what the distributions look like.

Does there exist an algorithm using less information?

What prior information is needed?

Remember that the KW-algorithm for selecting a single item uses the threshold

$$T = \frac{\mathbb{E}[\max_j X_j]}{2}.$$

Computing threshold requires full knowledge of the distributions X_i.

- Can be non-trivial depending on what the distributions look like.

Does there exist an algorithm using less information?

Turns out that it suffices to have one sample x_i from every X_i.

What prior information is needed?

Remember that the KW-algorithm for selecting a single item uses the threshold

$$T = \frac{\mathbb{E}[\max_j X_j]}{2}.$$

Computing threshold requires full knowledge of the distributions X_i.

- Can be non-trivial depending on what the distributions look like.

Does there exist an algorithm using less information?

Turns out that it suffices to have one sample x_i from every X_i.

Theorem (Rubinstein, Wang and Weinberg, 2020)

Suppose we have one sample x_i form every X_i, and let $T = \max_j x_j$. Selecting first element with $w_i \geq T$ gives prophet inequality with $\alpha = \frac{1}{2}$.

What prior information is needed?

Remember that the KW-algorithm for selecting a single item uses the threshold

$$T = \frac{\mathbb{E}[\max_j X_j]}{2}.$$

Computing threshold requires full knowledge of the distributions X_i.

- Can be non-trivial depending on what the distributions look like.

Does there exist an algorithm using less information?

Turns out that it suffices to have one sample x_i from every X_i.

Theorem (Rubinstein, Wang and Weinberg, 2020)

Suppose we have one sample x_i form every X_i, and let $T = \max_j x_j$. Selecting first element with $w_i \geq T$ gives prophet inequality with $\alpha = \frac{1}{2}$.

- Same guarantee as KW-algorithm.

What prior information is needed?

Remember that the KW-algorithm for selecting a single item uses the threshold

$$T = \frac{\mathbb{E}[\max_j X_j]}{2}.$$

Computing threshold requires full knowledge of the distributions X_i.

- Can be non-trivial depending on what the distributions look like.

Does there exist an algorithm using less information?

Turns out that it suffices to have one sample x_i from every X_i.

Theorem (Rubinstein, Wang and Weinberg, 2020)

Suppose we have one sample x_i form every X_i, and let $T = \max_j x_j$. Selecting first element with $w_i \geq T$ gives prophet inequality with $\alpha = \frac{1}{2}$.

- Same guarantee as KW-algorithm.
- Algorithms only using single sample from every X_i will be called single-sample algorithms.

Single-sample algorithms for matroid constraints

Azar, Kleinberg and Weinberg (2014) give single sample algorithms leading to constant-factor prophet inequalities for various matroid constraints.

Single-sample algorithms for matroid constraints

Azar, Kleinberg and Weinberg (2014) give single sample algorithms leading to constant-factor prophet inequalities for various matroid constraints.

- The high-level idea is to give a reduction to the secretary problem.

Single-sample algorithms for matroid constraints

Azar, Kleinberg and Weinberg (2014) give single sample algorithms leading to constant-factor prophet inequalities for various matroid constraints.

- The high-level idea is to give a reduction to the secretary problem.
- Samples are used to mimic “observation phase” (Phase I).

Single-sample algorithms for matroid constraints

Azar, Kleinberg and Weinberg (2014) give single sample algorithms leading to constant-factor prophet inequalities for various matroid constraints.

- The high-level idea is to give a reduction to the secretary problem.
- Samples are used to mimic “observation phase” (Phase I).
  - Slightly stronger, order-oblivious secretary algorithm is needed.

Single-sample algorithms for matroid constraints

Azar, Kleinberg and Weinberg (2014) give single sample algorithms leading to constant-factor prophet inequalities for various matroid constraints.

- The high-level idea is to give a reduction to the secretary problem.
- Samples are used to mimic “observation phase” (Phase I).
  - Slightly stronger, order-oblivious secretary algorithm is needed.
  - An example is the $\frac{1}{4}$-approximation we saw in Homework 3.

Single-sample algorithms for matroid constraints

Azar, Kleinberg and Weinberg (2014) give single sample algorithms leading to constant-factor prophet inequalities for various matroid constraints.

- The high-level idea is to give a reduction to the secretary problem.
- Samples are used to mimic "observation phase" (Phase I).
  - Slightly stronger, order-oblivious secretary algorithm is needed.
  - An example is the $\frac{1}{4}$-approximation we saw in Homework 3.

Theorem (Azar, Kleinberg and Weinberg, 2014 (informal))

Every order-oblivious α-approximation for the secretary problem (with uniform random arrivals) gives rise to a single-sample prophet inequality with factor α (for worst-case arrival order).

Single-sample algorithms for matroid constraints

Azar, Kleinberg and Weinberg (2014) give single sample algorithms leading to constant-factor prophet inequalities for various matroid constraints.

- The high-level idea is to give a reduction to the secretary problem.
- Samples are used to mimic “observation phase” (Phase I).
  - Slightly stronger, order-oblivious secretary algorithm is needed.
  - An example is the $\frac{1}{4}$-approximation we saw in Homework 3.

Theorem (Azar, Kleinberg and Weinberg, 2014 (informal))

Every order-oblivious α-approximation for the secretary problem (with uniform random arrivals) gives rise to a single-sample prophet inequality with factor α (for worst-case arrival order).

Reduction also works for graphic matroid algorithm from last week.

Single-sample algorithms for matroid constraints

Azar, Kleinberg and Weinberg (2014) give single sample algorithms leading to constant-factor prophet inequalities for various matroid constraints.

- The high-level idea is to give a reduction to the secretary problem.
- Samples are used to mimic "observation phase" (Phase I).
  - Slightly stronger, order-oblivious secretary algorithm is needed.
  - An example is the $\frac{1}{4}$-approximation we saw in Homework 3.

Theorem (Azar, Kleinberg and Weinberg, 2014 (informal))

Every order-oblivious α-approximation for the secretary problem (with uniform random arrivals) gives rise to a single-sample prophet inequality with factor α (for worst-case arrival order).

Reduction also works for graphic matroid algorithm from last week.

Corollary (AKW, 2014)

There is a single-sample $\alpha = \frac{1}{8}$ graphic matroid prophet inequality.

From single-sample prophets to secretaries

An algorithm (for adversarial arrival order σ) with samples x_i from X_i:

From single-sample prophets to secretaries

An algorithm (for adversarial arrival order σ) with samples x_i from X_i:

Preprocessing:

From single-sample prophets to secretaries

An algorithm (for adversarial arrival order σ) with samples x_i from X_i:

Preprocessing:

- Set $k = \frac{m}{2}$, and select uniformly at random k samples from $\{x_1, \ldots, x_m\}$.

From single-sample prophets to secretaries

An algorithm (for adversarial arrival order σ) with samples x_i from X_i:
Preprocessing:

- Set $k = \frac{m}{2}$, and select uniformly at random k samples from $\{x_1, \ldots, x_m\}$. Call the set of k samples

$$\{y_{j_1}, \ldots, y_{j_k}\}.$$

From single-sample prophets to secretaries

An algorithm (for adversarial arrival order σ) with samples x_i from X_i:
Preprocessing:

- Set $k = \frac{m}{2}$, and select uniformly at random k samples from $\{x_1, \ldots, x_m\}$. Call the set of k samples
$$\{y_{j_1}, \ldots, y_{j_k}\}.$$

Online:

From single-sample prophets to secretaries

An algorithm (for adversarial arrival order σ) with samples x_i from X_i:
Preprocessing:

- Set $k = \frac{m}{2}$, and select uniformly at random k samples from $\{x_1, \ldots, x_m\}$. Call the set of k samples
$$\{y_{j_1}, \ldots, y_{j_k}\}.$$

Online:

- For $i = 1, \ldots, m$, upon the arrival of $\sigma(i)$:

From single-sample prophets to secretaries

An algorithm (for adversarial arrival order σ) with samples x_i from X_i:

Preprocessing:

- Set $k = \frac{m}{2}$, and select uniformly at random k samples from $\{x_1, \ldots, x_m\}$. Call the set of k samples
$$\{y_{j_1}, \ldots, y_{j_k}\}.$$

Online:

- For $i = 1, \ldots, m$, upon the arrival of $\sigma(i)$:
  - If $\sigma(i) \in \{j_1, \ldots, j_k\}$, do nothing.

From single-sample prophets to secretaries

An algorithm (for adversarial arrival order σ) with samples x_i from X_i:

Preprocessing:

- Set $k = \frac{m}{2}$, and select uniformly at random k samples from $\{x_1, \ldots, x_m\}$. Call the set of k samples
$$\{y_{j_1}, \ldots, y_{j_k}\}.$$

Online:

- For $i = 1, \ldots, m$, upon the arrival of $\sigma(i)$:
  - If $\sigma(i) \in \{j_1, \ldots, j_k\}$, do nothing.
  - Otherwise, select $\sigma(i)$ if $w_i \geq \max\{y_{j_1}, \ldots, y_{j_k}\}$.

From single-sample prophets to secretaries

An algorithm (for adversarial arrival order σ) with samples x_i from X_i:

Preprocessing:

- Set $k = \frac{m}{2}$, and select uniformly at random k samples from $\{x_1, \ldots, x_m\}$. Call the set of k samples
$$\{y_{j_1}, \ldots, y_{j_k}\}.$$

Online:

- For $i = 1, \ldots, m$, upon the arrival of $\sigma(i)$:
  - If $\sigma(i) \in \{j_1, \ldots, j_k\}$, do nothing.
  - Otherwise, select $\sigma(i)$ if $w_i \geq \max\{y_{j_1}, \ldots, y_{j_k}\}$.

Theorem (AKW, 2014)

The above algorithm gives a single-sample prophet inequality with $\alpha = \frac{1}{4}$ for selecting one element.

From single-sample prophets to secretaries

An algorithm (for adversarial arrival order σ) with samples x_i from X_i:

Preprocessing:

- Set $k = \frac{m}{2}$, and select uniformly at random k samples from $\{x_1, \ldots, x_m\}$. Call the set of k samples

$$\{y_{j_1}, \ldots, y_{j_k}\}.$$

Online:

- For $i = 1, \ldots, m$, upon the arrival of $\sigma(i)$:
  - If $\sigma(i) \in \{j_1, \ldots, j_k\}$, do nothing.
  - Otherwise, select $\sigma(i)$ if $w_i \geq \max\{y_{j_1}, \ldots, y_{j_k}\}$.

Theorem (AKW, 2014)

The above algorithm gives a single-sample prophet inequality with $\alpha = \frac{1}{4}$ for selecting one element.

- Proof uses the fact that both (offline) sample x_i and (online) realization w_i come from the same distribution X_i.

Prophet inequalities for I.I.D. distributions

When all distributions X_i are the same

Better prophet inequalities (than $\alpha = \frac{1}{2}$) are possible when all distributions X_i are the same.

When all distributions X_i are the same

Better prophet inequalities (than $\alpha = \frac{1}{2}$) are possible when all distributions X_i are the same.

- The X_i are independent and identically distributed (I.I.D.).

When all distributions X_i are the same

Better prophet inequalities (than $\alpha = \frac{1}{2}$) are possible when all distributions X_i are the same.

- The X_i are independent and identically distributed (I.I.D.).

Theorem (Correa et al., 2017)

In case all the X_i are I.I.D. there exists a prophet inequality with $\alpha \approx 0.745$ and this is best possible.

When all distributions X_i are the same

Better prophet inequalities (than $\alpha = \frac{1}{2}$) are possible when all distributions X_i are the same.

- The X_i are independent and identically distributed (I.I.D.).

Theorem (Correa et al., 2017)

In case all the X_i are I.I.D. there exists a prophet inequality with $\alpha \approx 0.745$ and this is best possible.

The algorithm has access to the weights revealed so far, and the common distribution X.

When all distributions X_i are the same

Better prophet inequalities (than $\alpha = \frac{1}{2}$) are possible when all distributions X_i are the same.

- The X_i are independent and identically distributed (I.I.D.).

Theorem (Correa et al., 2017)

In case all the X_i are I.I.D. there exists a prophet inequality with $\alpha \approx 0.745$ and this is best possible.

The algorithm has access to the weights revealed so far, and the common distribution X. What is possible when X is unknown?

When all distributions X_i are the same

Better prophet inequalities (than $\alpha = \frac{1}{2}$) are possible when all distributions X_i are the same.

- The X_i are independent and identically distributed (I.I.D.).

Theorem (Correa et al., 2017)

In case all the X_i are I.I.D. there exists a prophet inequality with $\alpha \approx 0.745$ and this is best possible.

The algorithm has access to the weights revealed so far, and the common distribution X. What is possible when X is unknown?

Theorem (Correa et al., 2018)

In case the online algorithm only has access to weights revealed so far (but not to common distribution X), there is a prophet inequality with $\alpha = \frac{1}{e}$ and this is best possible.

Secretary prophet inequalities

Prophet secretary problems

In the prophet secretary model, the elements in $\{e_1, \ldots, e_m\}$

Prophet secretary problems

In the prophet secretary model, the elements in $\{e_1, \ldots, e_m\}$

- arrive in uniform random order

Prophet secretary problems

In the prophet secretary model, the elements in $\{e_1, \ldots, e_m\}$

- arrive in uniform random order
- with weight w_i drawn from known distribution X_i for $i = 1, \ldots, m$.

Prophet secretary problems

In the prophet secretary model, the elements in $\{e_1, \ldots, e_m\}$

- arrive in uniform random order
- with weight w_i drawn from known distribution X_i for $i = 1, \ldots, m$.

In this case, it is possible to obtain better results.

Prophet secretary problems

In the prophet secretary model, the elements in $\{e_1, \ldots, e_m\}$

- arrive in uniform random order
- with weight w_i drawn from known distribution X_i for $i = 1, \ldots, m$.

In this case, it is possible to obtain better results.

- These results apply to the general setting with possibly non-I.I.D. distributions

Prophet secretary problems

In the prophet secretary model, the elements in $\{e_1, \ldots, e_m\}$

- arrive in uniform random order
- with weight w_i drawn from known distribution X_i for $i = 1, \ldots, m$.

In this case, it is possible to obtain better results.

- These results apply to the general setting with possibly non-I.I.D. distributions

Theorem (Ehsani et al., 2018 (informal))

There is a secretary prophet inequality with $\alpha = 1 - \frac{1}{e} \approx 0.63$ *for selecting multiple elements under a matroid constraint.*

Prophet secretary problems

In the prophet secretary model, the elements in $\{e_1, \ldots, e_m\}$

- arrive in uniform random order
- with weight w_i drawn from known distribution X_i for $i = 1, \ldots, m$.

In this case, it is possible to obtain better results.

- These results apply to the general setting with possibly non-I.I.D. distributions

Theorem (Ehsani et al., 2018 (informal))

There is a secretary prophet inequality with $\alpha = 1 - \frac{1}{e} \approx 0.63$ for selecting multiple elements under a matroid constraint.

Theorem (Correa, Saona and Ziliotto, 2019)

There is a secretary prophet inequality with $\alpha = 1 - \frac{1}{e} + \frac{1}{27} \approx 0.669$ for selecting a single element.

Overview

Overview second part of course

Have seen various online selection problems and models.

Overview second part of course

Have seen various online selection problems and models.

Elements with unknown weights, but assumption on arrival order.

Overview second part of course

Have seen various online selection problems and models.

Elements with unknown weights, but assumption on arrival order.

- Secretary problem

Overview second part of course

Have seen various online selection problems and models.

Elements with unknown weights, but assumption on arrival order.

- Secretary problem
  - $\frac{1}{e}$-approximation.

Overview second part of course

Have seen various online selection problems and models.

Elements with unknown weights, but assumption on arrival order.

- Secretary problem
  - $\frac{1}{e}$-approximation.
- Online bipartite matching (nodes on one side arriving online).

Overview second part of course

Have seen various online selection problems and models.

Elements with unknown weights, but assumption on arrival order.

- Secretary problem
  - $\frac{1}{e}$-approximation.
- Online bipartite matching (nodes on one side arriving online).
  - $\frac{1}{e}$-approximation.

Overview second part of course

Have seen various online selection problems and models.

Elements with unknown weights, but assumption on arrival order.

- Secretary problem
  - $\frac{1}{e}$-approximation.
- Online bipartite matching (nodes on one side arriving online).
  - $\frac{1}{e}$-approximation.
- Matroid secretary problem.

Overview second part of course

Have seen various online selection problems and models.

Elements with unknown weights, but assumption on arrival order.

- Secretary problem
  - $\frac{1}{e}$-approximation.
- Online bipartite matching (nodes on one side arriving online).
  - $\frac{1}{e}$-approximation.
- Matroid secretary problem.
  - Open whether there is constant-factor approximation,

Overview second part of course

Have seen various online selection problems and models.

Elements with unknown weights, but assumption on arrival order.

- Secretary problem
  - $\frac{1}{e}$-approximation.
- Online bipartite matching (nodes on one side arriving online).
  - $\frac{1}{e}$-approximation.
- Matroid secretary problem.
  - Open whether there is constant-factor approximation,
  - or possibly $\frac{1}{e}$-approximation.

Overview second part of course

Have seen various online selection problems and models.

Elements with unknown weights, but assumption on arrival order.

- Secretary problem
  - $\frac{1}{e}$-approximation.
- Online bipartite matching (nodes on one side arriving online).
  - $\frac{1}{e}$-approximation.
- Matroid secretary problem.
  - Open whether there is constant-factor approximation,
  - or possibly $\frac{1}{e}$-approximation.
- Most algorithms can be turned into online strategyproof mechanisms for selling items to (unit-)demand bidders.

Overview second part of course

Have seen various online selection problems and models.

Elements with unknown weights, but assumption on arrival order.

- Secretary problem
  - $\frac{1}{e}$-approximation.
- Online bipartite matching (nodes on one side arriving online).
  - $\frac{1}{e}$-approximation.
- Matroid secretary problem.
  - Open whether there is constant-factor approximation,
  - or possibly $\frac{1}{e}$-approximation.
- Most algorithms can be turned into online strategyproof mechanisms for selling items to (unit-)demand bidders.

Known weight distributions of elements, but adversarial arrival order.

Overview second part of course

Have seen various online selection problems and models.

Elements with unknown weights, but assumption on arrival order.

- Secretary problem
  - $\frac{1}{e}$-approximation.
- Online bipartite matching (nodes on one side arriving online).
  - $\frac{1}{e}$-approximation.
- Matroid secretary problem.
  - Open whether there is constant-factor approximation,
  - or possibly $\frac{1}{e}$-approximation.
- Most algorithms can be turned into online strategyproof mechanisms for selling items to (unit-)demand bidders.

Known weight distributions of elements, but adversarial arrival order.

- Prophet inequality with $\alpha = \frac{1}{2}$ for selecting single element.

Overview second part of course

Have seen various online selection problems and models.

Elements with unknown weights, but assumption on arrival order.

- Secretary problem
  - $\frac{1}{e}$-approximation.
- Online bipartite matching (nodes on one side arriving online).
  - $\frac{1}{e}$-approximation.
- Matroid secretary problem.
  - Open whether there is constant-factor approximation,
  - or possibly $\frac{1}{e}$-approximation.
- Most algorithms can be turned into online strategyproof mechanisms for selling items to (unit-)demand bidders.

Known weight distributions of elements, but adversarial arrival order.

- Prophet inequality with $\alpha = \frac{1}{2}$ for selecting single element.
- Prophet inequality with $\alpha = \frac{1}{2}$ for matroid constraint.

Overview second part of course

Have seen various online selection problems and models.

Elements with unknown weights, but assumption on arrival order.

- Secretary problem
  - $\frac{1}{e}$-approximation.
- Online bipartite matching (nodes on one side arriving online).
  - $\frac{1}{e}$-approximation.
- Matroid secretary problem.
  - Open whether there is constant-factor approximation,
  - or possibly $\frac{1}{e}$-approximation.
- Most algorithms can be turned into online strategyproof mechanisms for selling items to (unit-)demand bidders.

Known weight distributions of elements, but adversarial arrival order.

- Prophet inequality with $\alpha = \frac{1}{2}$ for selecting single element.
- Prophet inequality with $\alpha = \frac{1}{2}$ for matroid constraint.
- Also saw some other models (e.g., single-sample settings).