Optimization and Uncertainty

Notes

Summer 2021

Martin Hoefer

Organizational

Lectures:

- Email: mhoefer@cs.uni-frankfurt.de
- Office: 115, R.M.S. 11-15, physical office hours postponed until further notice
- Lectures Tue/Thu every week, mostly writing on the board, recorded as videos
- All relevant text is in this document – only figures with examples are missing (I make them up on the fly in the lecture)
- Will provide additional material on the website
- This is a very new course, held for the very first time. This document is continously **expanded, updated and corrected** throughout the course.
- Please bear with me if things are not as smooth immediately as in other courses.

Exercises/Exams:

- Weekly exercise sheets, published on Tuesday of week i, due Tuesday of week $i + 1$,
- Turn it in via email, use **PDF format**.
- Great to discuss solutions, but write them up in **your own words**
- Returned and discussed in the next exercise session
- Exercise sessions online, live via Zoom
- Access data for the room provided in the beginning of the semester
- If you score $x\%$ of total number of exercise points, then
  If $50 \leq x < 75$, one grading step bonus for exam (e.g., 2.0 to 1.7, or 3.7 to 3.3)
  If $75 \leq x$, two grading steps bonus for exam (e.g., 2.0 to 1.3, or 3.3 to 2.7)
- Oral exams in August 2021

Contents

Chapter 1

Online Optimization

The central challenge of online algorithms is *uncertainty about the future*. In an online problem, the input is revealed gradually over time. An online algorithm must directly respond to the input pieces and make **immediate and irrevocable** decisions without knowing future parts of the input. In this section, we briefly introduce fundamental ideas and analysis techniques surrounding online problems.

Running example in this section is the Ski-Rental problem:

- There are n days, n known in advance, each day you want to go skiing.
- Only in the morining you realize if the ski resort is open on that day or not
- Each day that it is open, you can either rent equipment or buy it
- Rent is 1€ per day, buying costs B€, $B > 1$.
- If you buy, you don't have to rent anymore

Goal: Minimize the total cost of renting plus buying.

Consider the **Simple Algorithm**:
Rent until there are $B-1$ open days. On the B-th open day buy the equipment.

We denote by σ the input sequence (here: label open/closed for each of the n days), by $c(ALG(\sigma))$ the cost of an algorithm ALG for the problem, and by $c(OPT(\sigma))$ the optimal cost achievable on the input sequence σ.

Theorem 1. *The simple algorithm obtains a cost of at most $2-1/B$ times the optimal cost.*

Proof. Use a case analysis based on the (unknown) number of open days.

- σ has at most $B-1$ open days: Algorithm rents only, optimum rents only. Algorithm is optimal: $c(ALG(\sigma)) = c(OPT(\sigma))$
- σ has at least B open days: Algorithm rents $B-1$ days and buys, $c(ALG(\sigma)) = 2B-1$. Optimum buys on day 1, $c(OPT(\sigma)) = B$. Hence:

$$c(ALG(\sigma)) = 2B-1 = \left(\frac{2B-1}{B}\right) \cdot B = \left(2 - \frac{1}{B}\right) \cdot c(OPT(\sigma)).$$

□

We use the **competitive ratio** to analyze online algorithms. A deterministic algorithm ALG for a minimization problem is α**-competitive** if for *every input sequence* σ we have

$$c(ALG(\sigma)) \quad \leq \quad \alpha \cdot c(OPT(\sigma) + b,$$

where $b \geq 0$ is a constant independent of the input. Similarly, for a maximization problem, where we want to maxmize the valuation function v, we call ALG α**-competitive** if for *every input sequence* σ we have

$$v(ALG(\sigma)) \quad \geq \quad \frac{1}{\alpha} \cdot v(OPT(\sigma) - b.$$

If $b = 0$, we call ALG *strictly* α-competitive.

Observe that no deterministic algorithm can beat the guarantee of the simple algorithm.

Theorem 2. *There is no deterministic algorithm for* Ski-Rental *that is strictly* α*-competitive for any* $\alpha < 2 - 1/B$.

Proof. Consider any deterministic algorithm *ALG*. Clearly, the algorithm never needs to rent after it buys, so it first rents for a number of days and then buys. We concentrate on instances with $n = 2B$. Let $0 \leq \ell \leq n$ be the last open day on which the algorithm rents.

[Pic: Cost of algorithm, cost of optimum]

Case $\ell = n$:

- Consider an input sequence σ that has only open days.
- Algorithm never buys, has cost $c(ALG(\sigma)) = n = 2B$.
- Optimum buys right away, $c(OPT(\sigma)) = B$.
- Hence, for this sequence σ, the competitive ratio is

$$\frac{c(ALG(\sigma))}{c(OPT(\sigma))} = \frac{2B}{B} = 2.$$

Case $\ell < n$:

- Consider an input sequence σ with $\ell + 1$ open days.
- Algorithm rents until day ℓ, then buys: $c(ALG(\sigma)) = \ell + B$.
- Optimum chooses the minimum of renting and buying, $c(OPT(\sigma)) = \min\{\ell + 1, B\}$.
- Hence, for this sequence σ, the competitive ratio is lower bounded by

$$\frac{c(ALG(\sigma))}{c(OPT(\sigma))} = \frac{\ell + B}{\min\{\ell + 1, B\}} = \max\left(\frac{\ell + B}{\ell + 1}, \frac{\ell + B}{B}\right) \geq \frac{2B - 1}{B} = 2 - \frac{1}{B} \ .$$

Note: For the maximum, the first term is decreasing in ℓ, the second increasing in ℓ. Thus, the maximum is minimized when they are the same, i.e., when $\ell + 1 = B$.

Thus, for every algorithm, there is an input sequence such that the cost of the algorithm and the optimum for that input sequence differs by at least a factor $2 - 1/B$. □

This theorem seems to suggest that the simple algorithm is really the best algorithmic idea we can hope for – it is optimal in terms of competitive ratio. But how meaningful is this statement?

One issue is that we restricted attention to *deterministic* algorithms. What if the algorithm is randomized? Another issue is the type of worst-case analysis – we require that the ratio must hold *for each and every input sequence*, even the worst one (for that algorithm). What if there is more information about the input, e.g., when we have some stochastic distribution about the opening days? Randomization in algorithm and/or input can lead to much better results.

Let us briefly discuss a simple randomized online algorithm. It uses random coin flips to make decisions about the input. As a consequence, the algorithm has an expected cost for every input σ (expected over internal randomization in the algorithm).

A randomized algorithm for a minimization problem is **α-competitive** if for *every input sequence* σ we have

$$\mathbb{E}[c(ALG(\sigma))] \quad \leq \quad \alpha \cdot c(OPT(\sigma) + b,$$

where $b \geq 0$ is a constant independent of the input. Similarly, for a maximization problem, where we want to maxmize the valuation function v, we call ALG **α-competitive** if for *every input sequence* σ we have

$$\mathbb{E}[v(ALG(\sigma))] \quad \geq \quad \frac{1}{\alpha} \cdot v(OPT(\sigma) - b.$$

If $b = 0$, we call ALG *strictly* α-competitive.

Note that the simple algorithm is optimal for a small number of open days. If there are many open days, the algorithm buys way too late. Hence, a natural idea is to buy earlier with some probability. This means we have more cost for a small number of open days, but less cost if there are many open days. Overall, this turns out to be a profitable adjustment.

The **Randomized Simple Algorithm**: Flip a fair 50/50 coin.
If heads: Run the Simple Algorithm.
Otherwise: Rent for $\frac{3}{5} \cdot B$ open days, then buy.

Theorem 3. *The randomized simple algorithm for* Ski-Rental *is strictly* 11/6*-competitive.*

Proof. Use a case analysis on the number k of open days.

Case $k < \frac{3}{5}B$:
- Both algorithm and optimum rent only, $c(ALG(\sigma)) = c(OPT(\sigma)) = k$.

Case $k \geq B$:
- With probability 0.5, the algorithm buys after $\frac{3}{5}B$ rounds, with prob. 0.5 it buys after $B - 1$ rounds.
- Expected cost is $\mathbb{E}[c(ALG(\sigma))] = \frac{1}{2}\left(\frac{3}{5} \cdot B + B\right) + \frac{1}{2}\left(B - 1 + B\right) < \frac{9}{5} \cdot B$.
- Cost of the optmimum is $c(OPT(\sigma)) = B$. Competitive ratio is at most $9/5 = 1.8$.

Case $\frac{3}{5}B \leq k < B$:

- With probability 0.5, the algorithm buys after $\frac{3}{5}B$ rounds, with prob. 0.5 it rents for k rounds.
- Expected cost is $\mathbb{E}[c(ALG(\sigma))] = \frac{1}{2}\left(\frac{3}{5}\cdot B + B\right) + \frac{1}{2}\cdot k = \frac{4}{5}\cdot B + \frac{k}{2}$.
- Cost of optimum is $c(OPT(\sigma)) = k$. Competitive ratio is

$$\frac{\mathbb{E}[c(ALG(\sigma))]}{c(OPT(\sigma))} = \frac{4}{5}\cdot\frac{B}{k} + \frac{1}{2} \leq \frac{4}{5}\cdot\frac{5}{3} + \frac{1}{2} = \frac{11}{6} = 1.8\bar{3}$$

□

Better randomized algorithms for this problem use a more elaborate random choice for the buying time. By carefully adjusting the probabilities, one can get a competitive ratio of $(1 - 1/\mathsf{e})^{-1} \approx 1.58$, and this is best possible.

Theorem 4. *There is a randomized algorithm for* Ski-Rental *that is strictly* $\left(1 - \frac{1}{\mathsf{e}}\right)^{-1}$*-competitive, and no randomized algorithm is strictly* α*-competitive for any* $\alpha < \left(1 - \frac{1}{\mathsf{e}}\right)^{-1}$.

Finally, let us quickly touch upon randomization in the input (instead of the algorithm). A simple example to model a randomized input in Ski-Rental is an IID (independent, identically distributed) model: There is an opening probability $0 \leq q \leq 1$, which is known in advance. Every day, nature flips an independent, identical coin with probability q to determine if the resort can open or remains closed.

Due to randomization in the input, we now have a distribution $\mathcal{D}$ over input instances. The cost of the optimum is a random variable. For the expectation $\mathbb{E}_{\mathcal{D}}[c(OPT(\sigma))]$ we consider the optimal cost for every instance in the distribution, weighted by the probability that this instance arises, which is given by the distribution $\mathcal{D}$.

Then a randomized algorithm for a minimization problem is (strictly) α**-competitive** for *input distribution* $\mathcal{D}$ if we have

$$\mathbb{E}_{\mathcal{D}}[c(ALG(\sigma))] \quad \leq \quad \alpha \cdot \mathbb{E}_{\mathcal{D}}[c(OPT(\sigma)].$$

The expectation for ALG is both over randomization in the input as well as randomization in the algorithm, while for OPT it is only over randomization in the input.

Similarly, for a maximization problem, where we want to maxmize the valuation function v, we call ALG (strictly) α**-competitive** for *input distribution* $\mathcal{D}$ if we have

$$\mathbb{E}_{\mathcal{D}}[v(ALG(\sigma))] \quad \geq \quad \frac{1}{\alpha} \cdot \mathbb{E}_{\mathcal{D}}[v(OPT(\sigma)].$$

Later in the course, we will also discuss *optimal online algorithms*, i.e., an algorithm ALG* that given an input distribution $\mathcal{D}$ generates the smallest expected cost $\mathbb{E}[ALG^*(\sigma)]$. In fact, the IID model for Ski-Rental will turn out to be a Markov decision process, and an optimal algorithm for this process is at follows: If the first open day is on day t, then decide as follows: If $n - t + 1 \geq \left\lfloor \frac{B-1}{q} + 2 \right\rfloor$, then buy; otherwise rent throughout the sequence.

Note that the optimal online algorithm still suffers from uncertainty about the future – it has only probabilistic information about the open days in the future and optimizes the choice to minimize the expected cost. This is not to be confused with the optimum, which can always see the entire input instance completely in advance and makes optimal decisions for each instance in the support of $\mathcal{D}$.

Chapter 2

Online Algorithms with Random-Order Arrival

2.1 The Secretary Problem

In this chapter, we study classes of online problems, in which one tries to find the optimal time to stop. Usually, there is a sequence of options that arise sequentially, and the goal is to find the most profitable (feasible combination of) options. This goal is complicated by the fact that options expire quickly (e.g., investment options in financial markets, or partner options in dating markets), so usually a decision about option i has to be made before knowing the profit of subsequent options $i+1, i+2, \ldots$.

2.1.1 OnlineMax Problem

The basic problem: OnlineMax

- Suppose you go on a sequence of n dates, n is known.
- During date $i = 1, \ldots, n$ you get to know the quality of the person (a number $v_i > 0$)
- You must decide whether to accept/reject the person before going on the next date(s)
- Accept $\rightarrow$ person becomes your partner, sequence is over, no more dates.
- Reject $\rightarrow$ person is hurt and leaves, you go on $(i+1)$-st date (or stay single if $i = n$)
- **Online** problem: Decisions are **immediate and irrevocable**.

Goal: Maximize value of the accepted person.

[Pic: Sequence of Dates]

Online Algorithms and Approximation:

- Suppose ALG is an algorithm that computes a solution to an instance.
- Let $v(S) \geq 0$ be the value of a solution S.
- **Maximization Problem:** Find a solution with value **as large as possible**.

In OnlineMax...

- The number of dates n is a known parameter
- The input sequence σ composed of values $v_1, \ldots, v_n$ is unknown

Algorithm 1: Optimal Secretary

$s \leftarrow \lfloor n/e \rfloor$
// Sample phase
for rounds $t = 1, \dots, s$ **do** reject $\pi(t)$
// Accept phase
for *rounds* $t = s+1, \dots, n$ **do**
  if $v_{\pi(t)} > \max_{t'<t} v_{\pi(t')}$ or $t = n$ **then** accept $\pi(t)$ **else** reject $\pi(t)$

- A solution S corresponds to a single accepted person.
- We set $v(S) = v_i$ if $S = \{i\}$.

Here is a simple randomized algorithm ALG: Choose a round uniformly at random and accept the person in that round (independent of its value). Consider the expected value of the algorithm:

$$\mathbb{E}[v(ALG(\sigma))] = \sum_{t=1}^{n} \frac{1}{n} \cdot v_t \geq \frac{1}{n} \cdot \max_t v_t = \frac{1}{n} \cdot v(OPT(\sigma)) \ ,$$

i.e., the algorithm is (strictly) n-competitive. It turns out this trivial algorithm is essentially optimal. The following result will be proved later in the course.

Theorem 5. *Every randomized algorithm for the* OnlineMax *problem is* $\Omega(n)$*-competitive.*

This is devastating – the (rather stupid) simple randomized algorithm really appears to be the best algorithmic idea one can hope for. From a different perspective, this result highlights a problem of the type of worst-case analysis: We are unable to distinguish between stupid and more intelligent algorithmic approaches. Using a mild stochastic assumption on the uncertainty in the instance, such a distinction becomes possible.

2.2 Secretary Problem

In the Secretary **problem** we assume that dates arrive in **uniform random order**.

- The number of dates n is a known parameter
- Values $v_1, \dots, v_n$ are unknown (determined, say, by nature)
- For simplicity: We assume all values v_i **are distinct**
- Arrival order π is chosen uniformly at random
- In round t, you see person $\pi(t)$ with value $v_{\pi(t)}$ and must decide accept/reject

Theorem 6. *Algorithm 1 is* $(\mathsf{e} + o(1))$*-competitive for the* Secretary *problem.*

Proof. We denote by i^* the optimal person. We show that the algorithm **accepts** i^* **with probability at least** $1/\mathsf{e}$.

- Accept $i^* \iff$ two events happen:
  Event A_t: i^* comes in round $t \geq s+1$, and
  Event $R_{(1,t-1)}$: you rejected everyone before

- The probability that A_t happens is $1/n$. If A_t happens, then what is the conditional probability $\Pr[R_{(1,t-1)} \mid A_t]$?

[Pic: Arrival, two events]

No accept before?

- We denote by $[n] = \{1, \dots, n\}$ the set of all people. Consider the set $S \subseteq [n] \setminus \{i^*\}$ of people that arrive in rounds $1, \dots, t-1$. What about the best candidate i^*_S in S?
- i^*_S comes in a round $1, \dots, s \Rightarrow$ reject in all rounds $1, \dots, t-1$.
- i^*_S comes in a round $s+1, \dots, t-1 \Rightarrow$ accept in some round $s+1, \dots, t-1$.
- Hence: i^*_S comes in a round $1, \dots, s \iff R_{(1,t-1)}$
- Given any set S, it gets permuted in uniform random order over rounds $1, \dots, t-1$. Hence, the probability that i^*_S comes in a round $1, \dots, s$ is $s/(t-1)$

Overall, conditioned on A_t, for any set S of people arriving in the first $t-1$ rounds, we have $\Pr[R_{(1,t-1)} \mid A_t] = s/(t-1)$.

The probability of accepting i^* can be lower bounded by

$$\begin{aligned}
&\sum_{t=s+1}^{n} \Pr[A_t] \cdot \Pr[R_{(1,t-1)} \mid A_t] \quad = \quad \sum_{t=s+1}^{n} \frac{1}{n} \cdot \frac{s}{t-1} \quad = \quad \frac{s}{n} \cdot \sum_{t=s}^{n-1} \frac{1}{t} \\
\geq \quad & \frac{s}{n} \cdot \left(\int_{t=s}^{n} \frac{1}{t} \right) \quad = \quad \frac{s}{n} \cdot (\ln n - \ln s) \quad = \quad \frac{s}{n} \cdot \ln \frac{n}{s} \\
= \quad & \frac{\lfloor n/e \rfloor}{n} \cdot \ln \frac{n}{\lfloor n/e \rfloor} \quad \geq \quad \left(\frac{1}{e} - \frac{1}{n} \right) \cdot \ln \frac{n}{n/e} \quad = \quad \frac{1}{e} - \frac{1}{n} \ .
\end{aligned}$$

[Pic: Sum-to-Integral]

The algorithm always accepts i^* with probability at least $1/\mathsf{e} - 1/n$. Hence, the expected value of the algorithm is at least $(1/\mathsf{e} - 1/n) \cdot v_{i^*}$. The optimum always accepts i^*, so the value is v_{i^*}. The competitive ratio becomes at most

$$\frac{v_{i^*}}{\left(\frac{1}{\mathsf{e}} - \frac{1}{n}\right) \cdot v_{i^*}} = \frac{e}{1 - \mathsf{e}/n} = \mathsf{e} + \frac{\mathsf{e}^2}{n - \mathsf{e}} = \mathsf{e} + o(1).$$

□

Indeed, this algorithm is the **optimal** algorithm for the **ordinal** secretary problem, in which we want to maximize the probability of accepting the best person.

Theorem 7. *Algorithm 1 maximizes the probability to accept the best person.*

Proof. We again use i^* to denote the best person, and i_t the person arriving in round t. i_t is determined by the uniform random order. In each round t we define two events:

B_t: i_t ist the best person seen so far.
N_t: i_t ist not the best person seen so far.

Note:

$$\Pr[i_t = i^* \mid N_t] = 0. \qquad \Pr[i_t = i^* \mid B_t] = \frac{t}{n}.$$

Observations:

- If N_t, then i_t not the best one so far, so i_t cannot be i^*.
- If B_t, then i_t is the best one so far. When is the best one from the first t rounds also the globally best one? If and only if i^* comes in the first t rounds! The probability for this is t/n.
- When should we accept? If N_t, then we never accept!
- If B_t, then we should accept if the chances are low that a better person comes later on. This is true late in the process, but not early on.

An optimal algorithm **only uses information on n, t, B_t and N_t for the decision** in round t. Other information (e.g. about values seen so far) do not matter (why?).

Let us consider the following probabilities:

- v_t: probability we accept i^* in some round $t, t+1, \ldots, n$ after N_t happened in round t.
- u_t: probability we accept i^* in some round $t, t+1, \ldots, n$ after B_t happened in round t.

We optimize our algorithm and try to make all u_t and v_t as large as possible. We proceed **backwards in time** using a backwards iteration:

- Consider round $t = n$.
  If B_n happens, then $i_n = i^*$. We accept and get $u_n = 1$. If N_n happens, then we never get i^*. Hence, $v_n = 0$.

- Consider round $t = n - 1$.
  If B_{n-1}, then $\Pr[i_{n-1} = i^* \mid B_{n-1}] = \frac{n-1}{n}$.
  $\rightarrow$ if we accept, then we get i^* with prob. $(n-1)/n$.
  $\rightarrow$ if we reject, then $\Pr[B_n \mid B_{n-1}] = \frac{1}{n}$ and $\Pr[N_n \mid B_{n-1}] = \frac{n-1}{n}$. The probability that i^* comes in round n (and gets accepted there) becomes

$$v_n \cdot \frac{n-1}{n} + u_n \cdot \frac{1}{n} \ .$$

  Thus,

$$\begin{aligned} u_{n-1} &= \max\left(\frac{n-1}{n},\ v_n \cdot \frac{n-1}{n} + u_n \cdot \frac{1}{n}\right) \\ &= \frac{n-1}{n} \cdot \max\left(1,\ v_n + u_n \cdot \frac{1}{n-1}\right) \\ &= \frac{n-1}{n} \cdot \max\left(1,\ \frac{1}{n-1}\right) = \frac{n-1}{n} \ . \end{aligned}$$

  Hence, if B_{n-1}, we should accept in round $n-1$.

If we see N_{n-1}, then we certainly do not obtain i^* in round $n-1$. Hence, we should reject, since we might get i^* in the final round n. Note $\Pr[B_n \mid N_{n-1}] = \frac{1}{n}$ and $\Pr[N_n \mid N_{n-1}] = \frac{n-1}{n}$, and therefore

$$\begin{aligned} v_{n-1} &= v_n \cdot \frac{n-1}{n} + u_n \cdot \frac{1}{n} \\ &= \frac{1}{n} \\ &= \frac{n-1}{n} \cdot \left(\frac{1}{n-1}\right) \quad . \end{aligned}$$

- Consider round $t = n-2$.
  If we see B_{n-2}, then $\Pr[i_{n-2} = i^* \mid B_{n-2}] = \frac{n-2}{n}$.
  $\rightarrow$ if we accept, then we get i^* with prob. $(n-2)/n$.
  $\rightarrow$ if we reject, then $\Pr[B_{n-1} \mid B_{n-2}] = \frac{1}{n-1}$ and $\Pr[N_{n-1} \mid B_{n-2}] = \frac{n-2}{n-1}$. The probability that we can accept i^* in round $n-1$ or n becomes

  $$v_{n-1} \cdot \frac{n-2}{n-1} + u_{n-1} \cdot \frac{1}{n-1} \quad .$$

  Thus,

  $$\begin{aligned} u_{n-2} &= \max\left(\frac{n-2}{n},\ v_{n-1} \cdot \frac{n-2}{n-1} + u_{n-1} \cdot \frac{1}{n-1}\right) \\ &= \frac{n-2}{n} \cdot \max\left(1,\ v_{n-1} \cdot \frac{n}{n-1} + u_{n-1} \cdot \frac{n}{(n-1)(n-2)}\right) \\ &= \frac{n-2}{n} \cdot \max\left(1,\ \frac{1}{n-1} + \frac{1}{n-2}\right) = \frac{n-2}{n} \end{aligned}$$

  Hence, if B_{n-1}, we should accept in round $n-2$.

  If we see N_{n-2}, then we certainly do not obtain i^* in round $n-2$. Hence, we should reject, since we might get i^* in round $n-1$ or n. Note $\Pr[B_{n-1} \mid N_{n-2}] = \frac{1}{n-1}$ and $\Pr[N_{n-1} \mid N_{n-2}] = \frac{n-2}{n-1}$, and therefore

  $$\begin{aligned} v_{n-2} &= v_{n-1} \cdot \frac{n-2}{n-1} + u_{n-1} \cdot \frac{1}{n-1} \\ &= \frac{n-2}{n(n-1)} + \frac{1}{n} \\ &= \frac{n-2}{n} \cdot \left(\frac{1}{n-1} + \frac{1}{n-2}\right) \quad . \end{aligned}$$

- For round $t = n-3, n-4, n-5, \ldots$ we can show by induction:

  $$u_t = \frac{t}{n} \cdot \max\left(1, \frac{1}{n-1} + \frac{1}{n-2} + \ldots + \frac{1}{t}\right)$$

  $$v_t = \frac{t}{n} \cdot \left(\frac{1}{n-1} + \frac{1}{n-2} + \ldots + \frac{1}{t}\right)$$

In this way, we obtain an optimal algorithm:

- If N_t, we never want to accept the current person.
- If B_t for large t (late in time) such that

$$\frac{1}{n-1} + \frac{1}{n-2} + \ldots + \frac{1}{t} = \sum_{i=t}^{n-1} \frac{1}{i} < 1 \ ,$$

  it is optimal to accept the current (= best so far) person.
- If B_t for small t (early in time), then

$$\frac{1}{n-1} + \frac{1}{n-2} + \ldots + \frac{1}{t} = \sum_{i=t}^{n-1} \frac{1}{i} > 1 \ .$$

  Here it is optimal to reject and wait for later round and better people.
- The optimal threshold value r is given by

$$\sum_{i=s}^{n} \frac{1}{i} \geq 1 \qquad \text{and} \qquad \sum_{i=s+1}^{n} \frac{1}{i} \leq 1 \ .$$

  By using an integral to approximate the sum, one can show that $s = \lfloor n/e \rfloor$. Thus, the optimal algorithm is, indeed, Algorithm 1.

□

2.3 Secretary Matching

The Secretary problem seems neat, but a little abstract, and the random-order assumption a bit unrealistic. Still, the problem has received an enourmous attention in the literature since the first (rigorous) consideration in the 1960s.

Recently, there are many applications in **online advertising** and **sponsored search**, which nowadays is a multi-billion-dollar market. In this scenario, there is a search engine like Google and a set of advertisers, who want to show advertisements on search result pages. When a search query arrives, the engine assigns an ad on the result page. Depending on the query, showing a particular ad might be more or less valuable (to the advertiser, the company running the search engine, or the user).

This is essentially an online problem: Given an advertiser and an online sequence of search queries, find the best query to display the ad on the result page. Given the extreme amount of search queries per minute, random-order arrival is not unrealistic.

As a more elaborate approach for the search application, we study Secretary Matching:

- $G = (L \cup R, E)$ is a bipartite graph
- Right set R of nodes known in advance (advertisers)
- Left set L of nodes arriving in uniform random order (search queries)
- Each edge $e \in E$ has a value $v_e \geq 0$

Algorithm 2: Bipartite Secretary Matching

1 $s \leftarrow \lfloor n/e \rfloor$
  `// Sample phase`
2 $L_0 \leftarrow \emptyset$
3 **for** *rounds* $t = 1, \ldots, s$ **do** $L_t \leftarrow L_{t-1} \cup \{\ell_t\}$
  `// Accept phase`
4 $M \leftarrow \emptyset$
5 **for** *rounds* $t = s+1, \ldots, n$ **do**
6 $L_t \leftarrow L_{t-1} \cup \{\ell_t\}$
7 $E_t \leftarrow E \cap (L_t \times R)$
8 Compute a Max-Weight-Matching $M^{*,t}$ in $G_t = (L_t \cup R, E_t)$
9 **if** ℓ_t *is matched in* $M^{*,t}$ **then** $e_t \leftarrow (\ell_t, r) \in M^{*,t}$; **else** $e_t \leftarrow \emptyset$
10 **if** r *unmatched in* M **then** add e_t to M
11 **return** M

- Upon arrival, a node $\ell \in L$ reveals its' incident edges to R and their values

[Pic: Sponsored search, Bipartite Matching Model]

We solve Secretary Matching using Algorithm 2. It is an extension of the standard algorithm for the secretary problem. After the sampling phase, the algorithm computes in every round t $= s+1, \ldots, n$ an optimal matching $M^{*,t}$ in the graph G_t of all the arrived vertices and edges – independent of any matching edges chosen in previous rounds. The algorithm only cares about the current vertex ℓ_t and the matching edge $(\ell_t, r) \in M^{*,t}$. If ℓ_t is unmatched in $M^{*,t}$, the algorithm simply does nothing. Now if the vertex in $r \in R$ is still unmatched in M, then we add (ℓ_t, r) to M; otherwise, the algorithm does nothing.

[Pic: One round of the algorithm]

Theorem 8. *Algorithm 2 is* $(\mathsf{e} + o(1))$*-competitive for* Secretary Matching.

Consider any fixed round $t = s+1, \ldots, n$. For the sake of the analysis, assume that set L_t and vertex ℓ_t are the result of a **simulation:**

1. Draw L_t as a random subset from L. Each vertex $\ell \in L$ has $\Pr[\ell \in L_t] = t/n$.
2. Draw ℓ_t as a random vertex from L_t. Each vertex $\ell \in L_t$ has $\Pr[\ell = \ell_t] = 1/t$.

In this simulation, each subset of t nodes from L has the same probability to become L_t, and every vertex $\ell \in L$ has the same probability to become ℓ_t. As such, the simulation produces the same distribution for L_t and ℓ_t as our random-order permutation.

Now consider round t. The random edge e_t, as defined in line 9 of the algorithm, is the only edge we consider for addition to M in round t. Note that e_t might be an empty edge if ℓ_t is unmatched in $M^{*,t}$. How valuable is this e_t?

Lemma 1. *For every given round* $t = s+1, \ldots, n$*, we have* $\mathbb{E}[v(e_t)] \geq v(M^*)/n$.

Proof. We use the simulation:

- In step 1 of the simulation, we determine L_t. Note: This fully determines $M^{*,t}$!
- Edge $e = (\ell, r) \in M^{*,t}$ becomes $e_t \Leftrightarrow \ell$ is chosen to be ℓ_t in step 2 of the simulation.
- Hence, every edge $e \in M^{*,t}$ has probability $1/t$ to become e_t. This implies that e_t has "average" value of $M^{*,t}$

$$\mathbb{E}[v(e_t) \mid L_t] = \frac{1}{t} \cdot v(M^{*,t}) \ .$$

[Pic: simulation, choice of ℓ_t determines e_t]

Now how does the matching $M^{*,t}$ relate to the optimal matching M^*?

- Consider G_t. The set L_t of vertices in rounds $1, \ldots, t$ is a random sample from L.
- Hence, for each $\ell \in L$ we have $\Pr[\ell \in L_t] = t/n$.
- Let $M^t = E_t \cap M^* = \{(\ell, r) \in M \mid \ell \in L_t\}$ be the set of edges from the optimal matching that exist in G_t.
- For every $(\ell, r) \in M^*$ we have $\Pr[(\ell, r) \in M^t] = \Pr[\ell \in L_t] = t/n$. Hence,

$$\mathbb{E}[v(M^t)] = \frac{t}{n} \cdot v(M^*)$$

- Now, clearly, since $M^{*,t}$ is the optimal matching in G_t, it is better than M^t:

$$v(M^{*,t}) \geq v(M^t) \ .$$

- This implies that $\mathbb{E}[v(e_t)] \geq v(M^*)/n$:

$$\begin{aligned}\mathbb{E}[v(e_t)] &= \sum_{L_t \subseteq L} \frac{t}{n} \cdot \mathbb{E}[v(e_t) \mid L_t] = \sum_{L_t \subseteq L} \frac{t}{n} \cdot \frac{1}{t} \cdot v(M^{*,t}) \\ &\geq \frac{1}{t} \cdot \sum_{L_t \subseteq L} \frac{t}{n} \cdot v(M^t) = \frac{1}{t} \cdot \mathbb{E}[v(M^t)] = \frac{1}{t} \cdot \frac{t}{n} \cdot v(M^*) = \frac{v(M^*)}{n} \ .\end{aligned}$$

□

[Pic: M^* restricted to G_t vs. $M^{*,t}$]

Now every time we add edge e_t to M, we increase the value of M in expectation by $v(M^*)/n$. How likely is it that we can add e_t to M? We prove a general upper bound on this **success probability**. Our bound holds **for every realization of e_t, i.e., no matter which edge e_t turns out to be**.

Lemma 2. *For every round $t = s + 1, \ldots, n$ the success probability is at least $s/(t-1)$, independent of the realization of edge e_t.*

Proof. Consider the simulation and fix a choice for set L_t and vertex $\ell_t \in L_t$.

- This fixes $M^{*,t}$ and $e_t = (\ell_t, r)$, i.e., it also fixes $r \in R$.
- We bound the success probability $p_s(e_t, L_t)$ that a given edge e_t can be added to M in round t given that a particular set L_t arrives in the first t rounds.
- Consider the previous round $t - 1$. Let B_{t-1} be the event that $e_{t-1} = (\ell_{t-1}, r)$.

- We extend the simulation: Determine vertex ℓ_{t-1} by random draw from $L_{t-1} = L_t \backslash \{\ell_t\}$. Conditioned on L_t and ℓ_t, this is equivalent to getting ℓ_{t-1} from random-order arrival.
- Consider M^*_{t-1}. There is at most one vertex from L_{t-1} matched to r in M^*_{t-1}.
- Thus, B_{t-1} happens with probability at most $1/(t-1)$.
- Suppose it does not happen, i.e., condition on $\neg B_{t-1}$ and proceed:
  Determine ℓ_{t-2} by random draw from $L_{t-2} = L_t \setminus \{\ell_t, \ell_{t-1}\}$. Conditioned on L_t, ℓ_t, ℓ_{t-1} and not B_{t-1}, this is equivalent to getting ℓ_{t-2} from random-order arrival.
- Consider M^*_{t-2}. There is at most one vertex from L_{t-2} matched to r in M^*_{t-2}. Thus, B_{t-2} happens with probability at most $1/(t-2)$.
- The argument can be repeated for all rounds $k < t$.

[Pic: Backwards simulation of ℓ_t, ℓ_{t-1}, etc.]

If one of the events B_k happens for $k = s+1, \ldots, t-1$, then we add some edge incident to r into M in one of these rounds. Note that for $k \leq s$ we never add edges to M. Overall, e_t can be added to M if and only if none of the events B_k happens, for all $k = s+1, \ldots, t-1$. The success probability is

$$p_s(e_t, L_t) = \Pr\left[\bigwedge_{k=s+1}^{t-1} \neg B_k\right] \geq \prod_{k=s+1}^{t-1} \left(1 - \frac{1}{k}\right) = \frac{s}{s+1} \cdot \frac{s+1}{s+2} \cdots \frac{t-2}{t-1} = \frac{s}{t-1} \ .$$

□

Proof of Theorem 8. We now combine the previous lemmas to prove the theorem.

- By Lemma 1, the expected value of edge e_t in round t is $\mathbb{E}[v(e_t)] \geq v(M^*)/n$.
- By Lemma 2, for every realization of e_t and L_t, the success probability is at least $p_s(e_t, L_t) \geq s/(t-1)$.
- We denote by v_t the contribution of round t to $v(M)$. Combining the lemmas yields

$$\mathbb{E}[v_t] = \mathbb{E}[v(e_t) \cdot p_s(e_t, L_t)] = \mathbb{E}[v(e_t)] \cdot \frac{s}{t-1} \geq \frac{v(M^*)}{n} \cdot \frac{s}{t-1} \ .$$

- By linearity of expectation, the expected value of $v(M)$ can be bounded by

$$\mathbb{E}[v(M)] = \mathbb{E}\left[\sum_{t=s+1}^{n} v_t\right] = \sum_{t=s+1}^{n} \mathbb{E}[v_t] \geq v(M^*) \cdot \sum_{t=s+1}^{n} \frac{1}{n} \cdot \frac{s}{t-1} \ .$$

This leads to the same expression that we bounded in the end of the proof of Theorem 7. Repeating the analysis (using $v(M^*)$ instead of v_{i^*}) yields the same competitive ratio. □

2.4 Item Allocation in Markets

A fundamental problem in (Internet) markets is the allocation of indivisible goods and services to a set of agents or users. Given a number of item (access rights, hardware, etc.), which user should receive which item? A standard objective in this context is social welfare – maximize the sum of valuations of the users for their respective assigned bundle of items.

Can we obtain good algorithms to compute allocations of goods and services to users that maximize social welfare? What if users arrive online over time to the market?

Formally, we study the following ITEM ALLOCATION problem:

- There is a set L of n users and a set R of m items
- Each user $\ell \in L$ has a valuation $v_\ell(S) \geq 0$ for every subset $S \subseteq R$ of items.
- Allocation $\mathcal{S} = (S_1, \ldots, S_n)$ of goods to users with $S_i \subseteq R$ and $S_i \cap S_j = \emptyset$ for all $i \neq j$
- Goal: Find an allocation to maximize social welfare $SW(\mathcal{S}) = \sum_{\ell \in L} v_\ell(S_i)$.
- We use $\mathcal{S}^* = (S_1^*, \ldots, S_n^*)$ to denote an optimal allocation.

The model in the previous section can be seen as a special case of **unit-demand valuations** – each user in $\ell \in L$ wants to have at most one item (i.e., wants to be matched to at most one vertex from R). Here we assume that each user $\ell \in L$ may get assigned more than one item from R, depending on her valuation v_ℓ.

In general, ITEM ALLOCATION is an NP-hard problem, even in the offline variant where all users and items are known in advance. More severely, it is even NP-hard to approximate within any factor of $n^{1-\varepsilon}$, for every constant $\varepsilon > 0$.

We consider ITEM ALLOCATION as an online problem parametrized by the size d of the largest bundle of items that any user is interested in – no user wants more than d items. Moreover, we analyze the impact of more supply – what if every item is available in $b \geq 1$ copies?

ITEM ALLOCATION with bundles of size at most d and item multiplicity b

- For every user $\ell \in L$ we have $v_\ell(S_\ell) = 0$ whenever $|S_\ell| > d$
  (no user wants a bundle of size $> d$)
- For every item $r \in R$ there are $b \geq 1$ copies.

This problem can be formulated using the following integer linear program (ILP):

$$\begin{aligned}
\text{Maximize} \quad & \sum_{\ell \in L} \sum_{\substack{S \subseteq R \\ |S| \leq d}} x_{\ell,S} v_\ell(S) \\
\text{subject to} \quad & \sum_{\substack{S \subseteq R \\ |S| \leq d}} x_{\ell,S} \leq 1 \qquad \forall \ell \in L \\
& \sum_{\ell \in L} \sum_{\substack{S \subseteq R \\ |S| \leq d \\ r \in S}} x_{\ell,S} \leq b \qquad \forall r \in R \\
& x_{\ell,S} \in \{0,1\} \quad \forall \ell \in L, S \subseteq R, |S| \leq d.
\end{aligned} \tag{2.1}$$

Consider the linear relaxation:

- We replace the last constraints $x_{\ell,S} \in \{0,1\}$ by $0 \leq x_{\ell,S} \leq 1$.
- Thereby we obtain a linear relaxation with fractional $x_{\ell,S}$.
- This allows to obtain more social welfare in the optimum.

Algorithm 3: Item Allocation with Bundles of Size $\leq d$ and Item Multiplicity b

$s \leftarrow \lfloor n \cdot e(2d)^{1/b}/(1 + e(2d)^{1/b}) \rfloor$
`// Sample phase`
$L_0 \leftarrow \emptyset$
for *rounds* $t = 1, \ldots, s$ **do** $L_t \leftarrow L_{t-1} \cup \{\ell_t\}$ and $S_{\ell_t} \leftarrow \emptyset$
`// Accept phase`
for *rounds* $t = s + 1, \ldots, n$ **do**
  $L_t \leftarrow L_{t-1} \cup \{\ell_t\}$
  Solve LP (2.2), let $x^{*,t}$ be the optimal fractional solution
  Round S_{ℓ_t} to S with probability $x^{*,t}_{\ell_t,S}$, for every bundle $S \subseteq R$, $|S| \leq d$.
  if *in* $(S_{\ell_1}, \ldots, S_{\ell_t})$ *any item gets allocated more than* b *times* **then** $S_{\ell_t} \leftarrow \emptyset$
return $\mathcal{S} = (S_1, \ldots, S_n)$

- Let x^* an the optimal solution for the linear relaxation.
- $SW(x^*)$ is an upper bound for our optimal social welfare: $SW(x^*) \geq SW(\mathcal{S}^*)$.

Clearly, if we can design an approximation algorithm with a competitive ratio against $SW(x^*)$, then it also obtains at most this ratio against $SW(\mathcal{S}^*)$. But can we even compute x^* in polynomial time? This is not obvious, since the program has $n \cdot \binom{m}{d}$ coefficients and variables in the objective function. For simplicity, let us assume d is a constant[1]. Then the size of the program becomes polynomial $O(n \cdot \binom{m}{d}) = O(nm^d)$.

Algorithm 3 solves the online version of Item Allocation, in which users arrive sequentially in random order. It extends the idea of Algorithm 2 and rejects the first s users. Afterwards, in every round t it solves the linear relaxation of ILP (2.1) for the set L_t of users arrived until round t and all items R, i.e., it solves the linear program (LP)

$$\begin{aligned}
\text{Maximize} \quad & \sum_{\ell \in L_t} \sum_{\substack{S \subseteq R \\ |S| \leq d}} x_{\ell,S} v_\ell(S) \\
\text{subject to} \quad & \sum_{\substack{S \subseteq R \\ |S| \leq d}} x_{\ell,S} \leq 1 \qquad \forall \ell \in L_t \\
& \sum_{\ell \in L_t} \sum_{\substack{S \subseteq R \\ |S| \leq d \\ r \in S}} x_{\ell,S} \leq b \qquad \forall r \in R \\
& x_{\ell,S} \in [0,1] \quad \forall \ell \in L_t, S \subseteq R, |S| \leq d.
\end{aligned} \tag{2.2}$$

Since a feasible solution x must satisfy the first set of constraints, we can interpret the entries $x_{\ell,S}$ for a fixed user ℓ as a probability distribution over possible bundles $S \subseteq R$, $|S| \leq d$. The algorithm uses randomized rounding to determine a candidate bundle S_{ℓ_t}. If all items

[1] If d is non-constant, the program can also be solved in polynomial time under the condition that the valuations are given implicitly using an oracle that answers certain types of queries about the most preferred bundle.

in S_{ℓ_t} are still available (i.e., the bundles $S_{\ell_1}, \dots, S_{\ell_{t-1}}$ have not exhausted any $r \in S_{\ell_t}$), we include S_{ℓ_t} into the allocation; otherwise ℓ_t gets nothing.

Theorem 9. *Algorithm 3 is $O(d^{1/b})$-competitive for* Item Allocation *with bundles of size $\leq d$ and item multiplicity b.*

Consider the competitive ratio.

- It decreases as b grows larger. This is intuitive, since the more copies per item are available, the less severe is a "mistake" when we give an item to a suboptimal user.
- On the other hand, the ratio decreases as d grows smaller. Again, this is natural, since assigning smaller bundles decreases the effect on other users who might be excluded from the solution due to the lack of items. Also, note the hardness results for the general problem (with $d = n$) – hence, for large d we cannot hope for good performance guarantees.

Similar effects are present in the length of the sampling phase:

- If b grows larger, the expression $(2d)^{1/b}$ becomes smaller, and so does s. Clearly, if more copies are available, less waiting and sampling is needed, since the effect of a wrongly assigned item is much less severe.
- If d grows larger, $(2d)^{1/b}$ becomes larger, and so does s. Assigning larger bundles can be harmful when we have many remaining agents. The algorithm must be more careful and accumulates users in the sample, such that when it allocates an item, it has more substantial evidence that this decision is indeed a good idea.

We proceed to prove the theorem. The proof follows along the lines of the proof of Theorem 8: Study each fixed round $t = s + 1, \dots, n$ separately and prove

- a lower bound on the expected value of $\mathbb{E}(v_{\ell_t}(S_{\ell_t})]$ in terms of $SW(x^*)$
- a lower bound on the success probability that none of the items in S_{ℓ_t} is exhausted.

We here use the fractional optimum x^* as reference. By combining the bounds, we get a lower bound on the expected contribution of each round to $\mathbb{E}[SW(\mathcal{S})]$. By linearity of expectation, this implies a lower bound on $\mathbb{E}[SW(\mathcal{S})]$ and, thus, a bonud on the competitive ratio in terms of $SW(x^*)$.

For a fixed round t, we again use the simulation to mimic random-order arrival:

- Draw L_t as a random subset of L, then draw ℓ_t randomly from L_t.
- Remove ℓ_t from L_t to obtain L_{t-1}, then draw ℓ_{t-1} randomly from L_{t-1}
- Repeat to determine $\ell_{t-2}, \ell_{t-3}, \dots$

Lemma 3. *For every round $t = s + 1, \dots, n$ we have $\mathbb{E}[v_{\ell_t}(S_{\ell_t})] \geq SW(x^*)/n$.*

This lemma can be proved completely analogously as Lemma 1. We reiterate the argument for completeness.

Proof. Consider round t.

- Suppose we have chosen L_t. This also determines $x^{*,t}$ and $SW(x^{*,t})$.
- Now every $\ell \in L_t$ has probability $1/t$ to become ℓ_t.
- Then every set S then is chosen to be S_{ℓ_t} with probability $x^{*,t}_{\ell_t,S}$.

- This shows
$$\mathbb{E}[v_{\ell_t}(S_{\ell_t}) \mid L_t] = \sum_{\ell \in L_t} \frac{1}{t} \sum_{S \subseteq R} x^{*,t}_{\ell,S} \cdot v_\ell(S) = \frac{1}{t} \cdot SW(x^{*,t})$$

Now how large is $SW(x^{*,t})$?

- Let $x^t_{\ell,S} = x^*_{\ell,S}$ for $\ell \in L_t$ and $x^t_{\ell,S} = 0$ otherwise.
- x^t restricts the optimum x^* to entries for L_t. Note: x^t is a feasible solution for LP (2.2).
- For each $\ell \in L$ we have $\Pr[\ell \in L_t] = t/n$. Thus,
$$\mathbb{E}[SW(x^t)] = \sum_{\ell \in L} \sum_{S \subseteq R} x^t_{\ell,S} v_\ell(S) = \sum_{\ell \in L} \frac{t}{n} \sum_{S \subseteq R} x^*_{\ell,S} v_\ell(S) = \frac{t}{n} \cdot SW(x^*)$$
- Since $x^{*,t}$ is the optimal feasible solution to LP (2.2):
$$SW(x^{t,*}) \geq SW(x^t).$$
- As a consequence, we see that $\mathbb{E}[v_{\ell_t}(S_{\ell_t})] \geq SW(x^*)/n$:
$$\begin{aligned}\mathbb{E}[v_{\ell_t}(S_{\ell_t})] &= \sum_{L_t \subseteq L} \frac{t}{n} \cdot \mathbb{E}[v_{\ell_t}(S_{\ell_t}) \mid L_t] = \sum_{L_t \subseteq L} \frac{t}{n} \cdot \frac{1}{t} \cdot SW(x^{*,t}) \\ &\geq \frac{1}{t} \sum_{L_t \subseteq L} \frac{t}{n} \cdot SW(x^t) = \frac{1}{t} \cdot \mathbb{E}[SW(x^t)] = \frac{1}{t} \cdot \frac{t}{n} \cdot SW(x^*) = \frac{SW(x^*)}{n} \ .\end{aligned}$$

□

Lemma 4. *For every round $t = s+1, \ldots, n$, the success probability is at least*
$$1 - d \cdot \left(\frac{e(n-s)}{s}\right)^b \ ,$$
independent of the realization of bundle S_{ℓ_t}.

Proof. We have to bound the probability that all items are still available, i.e., each item $r \in S_{\ell_t}$ occurs at most $b-1$ times in the bundles $S_{\ell_1}, \ldots, S_{\ell_{t-1}}$.

- Fix a choice of L_t, the user ℓ_t , his chosen set S_{ℓ_t}.
- Consider an item $r \in S_{\ell_t}$. We again proceed backwards from round t to $s+1$.
- Consider the simulation. In every round $k \leq t-1$, once we have obtained L_k, we can assume that all remaining users $\ell \in L_k$ choose a tentative set S^k_ℓ according to the probabilities $x^{*,k}$.
- Then if a user ℓ is drawn from L_k to become ℓ_k, his set S^k_ℓ becomes S_{ℓ_k}.
- Indeed, since the rounding choice of S for each user and the random choice of ℓ_k are independent decisions, it doesn't matter if ℓ chooses his set before or after he is chosen to become ℓ_k.
- Now consider the probability that item r is in S_{ℓ_k}:
$$\Pr[r \in S_{\ell_k}] = \sum_{\ell \in L_k} \Pr[\ell = \ell_k] \cdot \Pr[r \in S^k_\ell] = \frac{1}{k} \cdot \sum_{\ell \in L_t} \sum_{S \subseteq R, r \in S} x^{*,k}_{\ell,S} \leq \frac{b}{k}$$
where the last inequality comes from the fact that $x^{*,t}$ fulfills the second set of constraints in LP 2.2.

Hence, in each round, the probability that r is chosen is only b/k. What is the probability that r is not chosen b times in the rounds $s+1, \ldots, t-1$?

$$\Pr[r \text{ in at least } b \text{ bundles}] \leq \sum_{\substack{C \subseteq \{s+1,\ldots,t-1\} \\ |C|=b}} \left(\prod_{k \in C} \frac{b}{k} \right) \leq \binom{t-1-s}{b} \left(\frac{b}{s} \right)^b \quad .$$

A general bound on binomial coefficients $(n/k)^k \leq \binom{n}{k} \leq (e \cdot n/k)^k$ comes in handy:

$$\Pr[r \text{ in at least } b \text{ bundles}] \leq \binom{t-1-s}{b} \left(\frac{b}{s} \right)^b \leq \left(\frac{e(t-1-s)}{s} \right)^b \leq \left(\frac{e(n-s)}{s} \right)^b \quad .$$

Note that $|S_{\ell_t}| \leq d$. The probability that at least one item in S_{ℓ_t} is chosen b times in previous rounds (i.e., the failure probability) is (by a union bound) at most $d \cdot \left(\frac{e(n-s)}{s} \right)^b$. The lemma follows. □

Proof of Theorem 9. We combine the previous lemmas to prove the theorem.

- By Lemma 3, the expected value of bundle S_{ℓ_t} in round t is $\mathbb{E}[v_{\ell_t}(S_{\ell_t})] \geq SW(x^*)/n$.
- By Lemma 4, for every realization of S_{ℓ_t} and L_t, the success probability is at least

$$1 - d \left(\frac{e(n-s)}{s} \right)^b \quad .$$

- We denote by v_t the contribution of round t to $SW(\mathcal{S})$. Combining the lemmas yields

$$\mathbb{E}[v_t] \geq \frac{SW(x^*)}{n} \cdot \left(1 - d \cdot \left(\frac{e(n-s)}{s} \right)^b \right) \quad .$$

By linearity of expectation, the expected value of $SW(\mathcal{S})$ can be bounded by

$$\mathbb{E}[SW(\mathcal{S})] = \sum_{t=s+1}^{n} \mathbb{E}[v_t] \geq SW(x^*) \cdot \frac{n-s}{n} \cdot \left(1 - d \cdot \left(\frac{e(n-s)}{s} \right)^b \right)$$

Substituting s into the expression, we notice that for large n, the rounding $\lfloor \ldots \rfloor$ in s only causes a negligible change overall:

$$\begin{aligned}
\mathbb{E}[SW(\mathcal{S})] &\geq SW(x^*) \cdot \left(\frac{1}{1 + e(2d)^{1/b}} \cdot \left(1 - d \cdot \left(\frac{e}{e(2d)^{1/b}} \right)^b \right) - o(1) \right) \\
&= SW(x^*) \cdot \left(\frac{1}{1 + e(2d)^{1/b}} \cdot \left(1 - \frac{d}{2d} \right) - o(1) \right) \\
&\geq SW(x^*) \cdot \left(\frac{1}{2 + 4ed^{1/b}} - o(1) \right) \\
&\geq SW(\mathcal{S}^*) \cdot \left(\frac{1}{2 + 4ed^{1/b}} - o(1) \right) \quad .
\end{aligned}$$

This proves the theorem. □

Chapter 3

Online Algorithms with Distributions

3.1 Prophet Inequalities

Let us reconsider the OnlineMax problem. Here we consider a slightly different and, arguably, more natural form of uncertainty. Instead of completely unknown values and random order, we now have **probability distributions** $\mathcal{D}_1, \ldots, \mathcal{D}_n$ for the rounds. The value of the person in round t is **drawn independently from** $\mathcal{D}_t$.

The basic challenge is the same: Should we accept a person now or wait for a better one? In random order models, a sample phase was useful to gain conditional information about future rounds. Now we have direct stochastic information about the future rounds.

The consideration of this scenario is older than competitive analysis, and the terminology is often slightly different: The offline optimum OPT is called a "prophet" who knows the future. The algorithm ALG is a gambler who wants to approximate the expected value of the prophet. Hence, we can see the definition of competitive ratio as a "prophet inequality", relating the performances of gambler (algorithm) and prohpet (optimum).

In the Prophet **problem** we assume that

- The number of dates n is a known parameter
- Distribution $\mathcal{D}_t$ is known for each round $t = 1, \ldots, n$ (no random order)
- Value v_t drawn independently from $\mathcal{D}_t$
- In round t, you see realization v_t. Decide accept/reject before seeing the next value(s)

3.1.1 Independent Distributions

Consider Algorithm 4. Instead of sampling, it computes τ as half of the expected value of OPT. It uses τ as acceptance threshold and accepts the first person with value at least τ. We will show that this yields at least τ as expected value of the algorithm.

This is noteworthy, since there might be a chance that the algorithm accepts nobody. Then, however, the result implies that there must also be a good probability that accepted people are significantly better than τ.

Theorem 10. *Algorithm 4 is* 2*-competitive for the* Prophet *problem.*

Proof. Consider the expected value of Algorithm 4.

Algorithm 4: Prophet Approximation

```
Compute the expected optimum value: v* ← E[max_{t=1}^n v_t]
Set τ = v*/2
for rounds t = 1, ..., n do
    if v_t ≥ τ then accept person t; else reject person t
```

- q is the probability that nobody is accepted: $q = \Pr[v_1 < \tau \wedge \ldots \wedge v_n < \tau]$
- Let A_t be the event that the algorithm accepts in round t.
- These events are mutually disjoint.
- For each t, we define $u_t = v_t - \tau$ when A_t happens, and $u_t = 0$ otherwise.
- The expected value of the algorithm is

$$\sum_{t=1}^{n} \mathbb{E}[v_t \mid A_t] \cdot \Pr[A_t] = \sum_{t=1}^{n} \mathbb{E}\left[u_t + \tau \mid A_t\right] \cdot \Pr[A_t] = \sum_{t=1}^{n} \mathbb{E}\left[u_t \mid A_t\right] \cdot \Pr[A_t] + \sum_{t=1}^{n} \tau \cdot \Pr[A_t]$$
$$= \sum_{t=1}^{n} \mathbb{E}\left[u_t\right] + (1-q)\tau \tag{3.1}$$

An economic interpretation:

- τ is the price of our car
- Every round t a person with value v_t comes up
- The first one that has value $v_t \geq \tau$ buys the car. Buyer t's utility is v_t.
- Our revenue is τ if anyone buys, so expected revenue is $(1-q)\tau$

We use $\mathbf{1}_x$ to denote 0/1 indicator variables that are 1 if and only if x is true. Now for every round t, we have

$$u_t = \max\{v_t - \tau, 0\} \cdot \mathbf{1}_{v_1 < \tau \wedge \ldots \wedge v_{t-1} < \tau},$$

By independence,

$$\begin{aligned}
\mathbb{E}[u_t] &= \mathbb{E}[\max\{v_t - \tau, 0\} \cdot \mathbf{1}_{v_1 < \tau \wedge \ldots \wedge v_{t-1} < \tau}] \\
&= \mathbb{E}[\max\{v_t - \tau, 0\}] \cdot \Pr[v_1 < \tau \wedge \ldots \wedge v_{t-1} < \tau] \\
&\geq \mathbb{E}[\max\{v_t - \tau, 0\}] \cdot q \ ,
\end{aligned}$$

Now

$$\begin{aligned}
\sum_{t=1}^{n} \mathbb{E}\left[u_t\right] &\geq \sum_{t=1}^{n} \mathbb{E}\left[\max\{v_t - \tau, 0\}\right] \cdot q && \text{shown above} \\
&= \mathbb{E}\left[\sum_{t=1}^{n} \max\{v_t - \tau, 0\}\right] \cdot q && \text{linearity of expectation} \\
&\geq \mathbb{E}\left[\max_{t=1}^{n} \max\{v_t - \tau, 0\}\right] \cdot q && \text{since } \sum_i x_i \geq \max_i x_i \text{ when all } x_i \geq 0. \\
&\geq \mathbb{E}\left[\max_{t=1}^{n} v_t - \tau\right] \cdot q && \text{adding negative values when all } v_t < \tau
\end{aligned}$$

$$= \left(\mathbb{E}\left[\max_{t=1}^{n} v_t\right] - \tau\right) \cdot q \qquad \text{linearity of expectation (again)}$$
$$= (v^* - \tau) \cdot q$$

where the second line uses linearity of expectation, and the third line uses $\sum_i x_i \geq \max_i x_i$ for all non-negative numbers $x_i \geq 0$.
We now use this bound in (3.1). We plug in our choice $\tau = v^*/2$ and see that the expected value of the algorithm is at least

$$\sum_{t=1}^{n} \mathbb{E}[u_t] + (1-q)\tau \geq (v^* - \tau) \cdot q + (1-q)\tau = (v^* - v^*/2) \cdot q + (1-q)v^*/2$$
$$= v^*/2 = \tau,$$

no matter what q is. □

Note that this guarantee cannot be improved in general.

Example 1. Suppose you have $n = 2$ rounds. In the first round, $v_1 = 1$ with probability 1. In the second round, $v_2 = 0$ with probability $1 - \varepsilon$, and $v_2 = 1/\varepsilon$ with probability $\varepsilon > 0$. Intuitively, in the first round the gambler gets 1€ for sure. In the second round, there is a tiny chance one wins a huge amount in the lottery, but usually one gets nothing. The gambler can decide to take the 1€ or wait. No matter what he does, however, the expected value is 1.

In contrast, the prophet knows in advance whether or not there is a lottery win in round 2. In that case (happens with probability ε) she waits for round 2. Otherwise (happens with probability $1 - \varepsilon$) she takes the 1€ in round 1. The expected value of the prophet is, thus, $1/\varepsilon \cdot \varepsilon + 1 \cdot (1 - \varepsilon) = 2 - \varepsilon$. Hence, gambler and prophet differ by a factor that is arbitrarily close to 2. ■

3.1.2 IID

We show an improved result for the IID scenario, when the value of each person is drawn i.i.d. (independent, identically distributed) from the same distribution $\mathcal{D}$.

Consider Algorithm 5. The $(1-1/n)$**-quantile** is the value such that $\Pr[v_1 \leq t] = 1-1/n$. The algorithm collects all values of at most the $(1 - 1/n)$-quantile in R and rejects them. Conversely, it accepts the first person with a value above the $(1 - 1/n)$-quantile.

Since $\mathcal{D}$ is discrete, there can be (at most) one value r', such that $\Pr[v_1 < r'] < 1 - 1/n$ and $\Pr[v_1 \leq r'] > 1 - 1/n$. When such an r' exists, there is no $(1 - 1/n)$-quantile. Then the algorithm internally gives any person with value r' a random label h or l. This splits r' into two values (r', h) and (r', l), where (r', h) should be accepted and (r', l) rejected. The probability for (r', l) in line 10 is chosen exactly such that $\Pr[(v_1 < r') \vee (v_1 = (r', l))] = 1 - 1/n$. In this way, (r', l) acts as the $(1 - 1/n)$-quantile.

[Pic: Discrete distribution, $(1 - 1/n)$-quantile, split of value r' into two values]

The algorithm has a competitive ratio of $(1 - 1/\mathsf{e})^{-1} \approx 1.58$.

Algorithm 5: IID-Prophet Approximation

Notation: $q_r = \Pr[v_1 = r]$.
Sort support $S_{\mathcal{D}} = \{r \mid q_r > 0\}$ in non-decreasing order
$x \leftarrow 0$, $i \leftarrow 0$, $R \leftarrow \emptyset$
while $i < |S_{\mathcal{D}}|$ *and* $x < 1 - 1/n$ **do**
  $x \leftarrow x + q_{r_i}$
  if $x \leq 1 - 1/n$ **then** $R \leftarrow R \cup \{r_i\}$; **else** $r' \leftarrow r_i$
  i++
for *rounds* $t = 1, \ldots, n$ **do**
  if $v_t \in R$ **then** reject person t
    else if $v_t = r'$ **then** reject person t with probability $(1 - 1/n - (x - q_{r'}))/q_{r'}$
    else accept person t

Theorem 11. *Algorithm 5 is* $(1-1/\mathsf{e})^{-1}$*-competitive for the* Prophet *problem with identical distributions.*

Proof. Consider the event E that a person is accepted by the algorithm.

- Values are drawn independently across rounds, the labels l, h for r' are also assigned independently each round.
- The probability that a person is rejected in round i is

$$\begin{aligned}\Pr[v_t \in R \cup \{(r', l)\}] &= \sum_{r \in R} q_r + q_{r'} \cdot \frac{1 - 1/n - (x - q_{r'})}{q_{r'}} \\ &= \sum_{r \in R} q_r + 1 - \frac{1}{n} - \left(\sum_{r \in R} q_r\right) = 1 - \frac{1}{n} \ .\end{aligned}$$

- Hence, $\Pr[E] = 1 - \Pr[\neg E] = 1 - \prod_{t=1}^{n} \left(1 - \frac{1}{n}\right)^n \geq 1 - \frac{1}{e}$.

Condition on E, and suppose $k \geq 1$ persons are above the $(1 - 1/n)$-quantile.

- Let F be the set of k rounds with persons above the $(1 - 1/n)$-quantile.
- We accept the person in the first round from F. Is this person **special**?
- Does the condition $t = \min\{j \in F\}$ change the expected value of v_t? Or is it just $\mathbb{E}[v \mid (v > r') \vee (v = (r', h))] = \mathbb{E}[v_t \mid v_t \text{ above } (1 - 1/n)\text{-quantile}]$?

Simulation: Draw n values from $\mathcal{D}$ and assign them in random order to the n rounds.

- The permutation has no effect on the distribution or the independence.
- First value above the $(1 - 1/n)$-quantile is a uniform random one from the k values.
- Hence: First person above the the $(1 - 1/n)$-quantile is **not special**!
- Any accepted person has expected value $\mathbb{E}[v_t \mid v_t \text{ above } (1 - 1/n)\text{-quantile}]$.

Consider the distribution of the optimal person.[1]

[1] If several persons have optimal value, break ties uniformly at random: Label a random person as optimal and the rest suboptimal.

- By simulation: Each round the person is optimal with probability $1/n$.
- Both events "v_t optimal" and "v_t above $(1-1/n)$-quantile" have probability of $1/n$ in each round.
- Above the $(1-1/n)$-quantile are the *best* realizations that make up a probability mass of $1/n$. Hence,

$$\mathbb{E}[v_t \mid v_t \text{ above } (1-1/n)\text{-quantile}] \geq \mathbb{E}[v_t \mid v_t \text{ optimal}] \ .$$

In case of event E, the accepted person has expected value better than the expected value of the optimum. However, event E occurs only with a probability of $1-(1-1/n)^n \geq 1-1/\mathsf{e}$. □

This is not the best algorithm for identical distributions. If $\mathcal{D}$ is continuous, there is an algorithm with a ratio approximately $1/0.745 = 1.342$, and there is a distribution for which this ratio cannot be improved.

3.2 Markov Decision Processes

The OnlineMax problem with distributions is a special case of a general framework of stochastic optimization problems. For exposition, our running example in this section will be a slightly simpler PrizeCollection problem:

- There are n envelopes. With probability $p_i \in [0,1]$, envelope i contains a prize of value $v_i > 0$. Otherwise, it is empty and has value 0.
- You open all envelopes in arbitrary order. You **keep all prizes** as long as you do **not open an empty** envelope.
- Once you open an empty envelope, you **stop earning prizes** from **future** envelopes.

Intuitively, starting with the envelopes that contain highest prizes or largest expected value seems like a good idea. However, there are examples where this strategy fails miserably: Suppose we have one envelope with $v_1 = 1000$ and $p_1 = 0.01$, and then 99 others with $v_i = p_i = 1$ for $i = 2, \ldots, 100$. If we open envelope 1 first, then with probability 0.99 the evenlope is empty and we get nothing – whereas with probability 0.01 we can continue and collect all prizes of total value 1099. Overall, this yields an expected value of 10.99. Clearly, the optimal strategy here would be to open only the envelopes $2, \ldots, 100$, by which we secure a value of 99.

PrizeCollection is an example of a **Markov Decision Process (MDP)**, where we have

- a set $\mathcal{S}$ of states, with an initial state $s_{init} \in \mathcal{S}$, and a set $\mathcal{A}$ of actions,
- a reward $r_a(s)$ for taking action a in state s, for all $a \in \mathcal{A}$, $s \in \mathcal{S}$, and
- a probability $p_a(s, s')$ to move from s to s' after taking action a, for all $a \in \mathcal{A}$, $s, s' \in \mathcal{S}$.

In round 1 we start in state $s_1 = s_{init}$. We choose some action a and receive reward $r_a(s_1)$. After getting the reward, we transition to some other state s_2 drawn independently at random according to $p_a(s_1, \cdot)$. Then we choose the next action, and so on.

Some observations:

- State s_t only depends on state and action in round t, but not on other parts of the history – this property of the process is called *Markovian*.

- Rewards might be stochastic (as in our example), then $r_a(s)$ is the *expected* reward.
- The model generalizes finite state automata (when the $p_a(s, s')$ are either 0 or 1) and Markov chains (when there is only a single action).

PrizeCollection as MDP:

- Let $[n] = \{1, \ldots, n\}$ be the set of envelopes. We have $\mathcal{S} = 2^{[n]} \cup \{\text{Empty}\}$ and $\mathcal{A} = [n]$.
- State $s \subseteq [n]$ means exactly the envelopes from s are open, all are non-empty.
- State Empty means there is at least one open empty envelope.
- Starting state is $s_{init} = \emptyset$.
- Action $a \in \mathcal{A}$ means you open envelope a

Rewards and Transitions:

- Reward $r_a(s) = v_a \cdot q_a$ if $a \notin s$: open new envelope a and get the expected reward.
- $p_a(s, s \cup \{a\}) = q_a$. If new envelope a contains a prize, move to $s \cup \{a\}$ and continue.
- $p_a(s, \text{Empty}) = 1 - q_a$: If new envelope a is empty, move to Empty.
- Wait a second: You always get the reward $r_a(s)$ for new envelope s, no matter whether it contains a prize or is empty? Yes, but only $v_a \cdot q_a$! Getting v_a when the envelope contains a prize and 0 otherwise has the same expectation as always getting $v_a \cdot q_a$.
- Reward $r_a(\text{Empty}) = 0$ and transition $p_a(\text{Empty}, \text{Empty}) = 1$, for all $a \in \mathcal{A}$. Once in Empty, you stay there and get no further reward.
- In MDPs we can play any action in any state. We forbid "re-opening" an envelope by rewards: $r_a(s) = -\infty$ and $p_a(s, s) = 1$, for $a \in s$.

[Pic: MDP for PrizeCollection states, actions, transitions]

3.2.1 Optimal Policies

In general, we could move through an MDP for unbounded time. We here focus on MDPs with **finite time horizon** with a finite number of $T \in \mathbb{N}$ rounds.

A (deterministic) **policy** π assigns each sequence of states $s_1, \ldots, s_{t-1}$ a single action $a \in \mathcal{A}$.

- Running policy π we see random sequences of states $s_1^\pi, \ldots, s_T^\pi$ and actions $a_1^\pi, \ldots, a_T^\pi$
- Expected reward of policy π is

$$V(\pi, s_{init}, T) = \mathbb{E}\left[\sum_{t=1}^{T} r_{a_t^\pi}(s_t^\pi)\right]$$

- Since they are deterministic, there are at most $\prod_{t=1}^{T} |\mathcal{S}|^t \times |\mathcal{A}|$ different policies. Hence, there is an **optimal policy** π^* with $V(\pi^*, s_{init}, T) = \max_\pi V(\pi, s_{init}, T)$.
- Notation $V^*(s_{init}, T) = V(\pi^*, s_{init}, T)$.

A particularly nice property of a policy is when it is **Markovian**. Then the choice of action in round t depends **only on** s_{t-1} **and on** t but **not on the rest of the history** $s_1, \ldots, s_{t-2}$. It turns out that there is at least one optimal policy that is also Markovian! Moreover, this policy can be computed in polynomial time.

Theorem 12. *For MDPs with finite time horizon, there is an optimal policy π^* that is Markovian. It can be computed in time $O(T \cdot |\mathcal{S}|^2 \cdot |\mathcal{A}|)$.*

Proof. Consider any optimal policy π with $V(\pi, s_{init}, T) = V^*(s_{init}, T)$. Since a_1^π is deterministic,

$$\begin{aligned} V(\pi, s_{init}, T) &= r_{a_1^\pi}(s_{init}) + \mathbb{E}\left[\sum_{t=2}^{T} r_{a_t^\pi}(s_t^\pi)\right] \\ &= r_{a_1^\pi}(s_{init}) + p_{a_1^\pi}(s_{init}, s') \cdot \mathbb{E}\left[\sum_{t=2}^{T} r_{a_t^\pi}(s_t^\pi) \;\middle|\; s_2^\pi = s'\right] \end{aligned}$$

Consider the last term on the right-hand side:

- Markovian: Rewards and transitions in rounds $2, \ldots, T$ do not depend on s_{init} and a_1^π.
- Consider choices of π in rounds $2, \ldots, T$ when $s_2 = s'$. Denote this subpolicy by $\pi_{2,s'}$.
- We can use $\pi_{2,s'}$ in another MDP that runs for $T-1$ rounds and starts in s'.
- Since the process is Markovian, $V(\pi_{2,s'}, s', T-1) = \mathbb{E}\left[\sum_{t=2}^{T} r_{a_t^\pi}(s_t^\pi) \;\middle|\; s_2^\pi = s'\right]$
- Now consider an optimal policy π' for that other MDP:

$$\mathbb{E}\left[\sum_{t=2}^{T} r_{a_t^\pi}(s_t^\pi) \;\middle|\; s_2^\pi = s'\right] = V(\pi_{2,s'}, s', T-1) \leq V(\pi', s', T-1) = V^*(s', T-1)$$

- We can also use π' instead of $\pi_{2,s'}$ in to play rounds $2, \ldots, T$ in the original MDP.
- Since the process is Markovian, $\mathbb{E}\left[\sum_{t=2}^{T} r_{a_t^{\pi'}}(s_t^{\pi'}) \;\middle|\; s_2^{\pi'} = s'\right] = V^*(s', T-1)$
- But π is an optimal policy, so this substitution must give no improvement, and hence

$$\mathbb{E}\left[\sum_{t=2}^{T} r_{a_t^\pi}(s_t^\pi) \;\middle|\; s_2^\pi = s'\right] = V^*(s', T-1)$$

Applying this argument recursively, for any round $t \geq 1$, we can assume that in rounds $t, \ldots, T$ an optimal policy for an MDP with $T-t+1$ rounds starting from state s_t is played. As such, there is an optimal policy where decisions depend only on the state s_t and the round t. The recusion for the optimal reward $V^*(s, T)$ is

$$V(s, T) = \max_{a \in \mathcal{A}} \left(r_a(s) + \sum_{s' \in \mathcal{S}} p_a(s, s') \cdot V^*(s', T-1) \right) \tag{3.2}$$

Computing this value and tracking the optimal actions choices can be done by dynamic programming in time $O(T \cdot |\mathcal{S}|^2 \cdot |\mathcal{A}|)$. □

3.2.2 Examples for Optimal Policies

PrizeCollection. To visualize the dynamic program, consider the following simple example. There are two rounds with $v_1 = 1000$, $q_1 = 0.01$ and $v_2 = q_2 = 1$. Clearly, we want to open envelope 2 before 1, with expected reward 1+10 = 11. Computing the dynamic program yields the following table (columns are states, rows are values of T):

T	$\emptyset$	$\{1\}$	$\{2\}$	$\{1,2\}$	EMPTY
0	0	0	0	**0**	**0**
1	10	**1**	**10**	$-\infty$	**0**
2	**11**	$-\infty$	$-\infty$	$-\infty$	0

Only the bold entries must be evaluated when invoking the recursion from $V(\emptyset, 2)$ with allowed actions. For the final step, we compare two options: Open envelope 1 first, which gives $10 + 0.01 \cdot 1 + 0.99 \cdot 0 = 10.01$; or envelope 2 first, which yields $1 + 1 \cdot 10 = 11$.

The direct approach suffers from the need to deal with more than 2^n many states. With a little more insight, we discover an optimal policy with much simpler structure:

- Optimal policy π^* defines an order in which envelopes are opened.
- Suppose in this order envelope i comes directly before j.
- Let s be the set of envelopes opened before i. Apply (3.2) twice to get

$$\begin{aligned} V^*(s,T) = V(\pi^*, s, T) &= v_i \cdot q_i + q_i \cdot V^*(s \cup \{i\}, T-1) \\ &= v_i \cdot q_i + q_i(v_j \cdot q_j + q_j \cdot V^*(s \cup \{i,j\}, T-2)) \end{aligned}$$

- We could also switch the order of j and i and keep the rest of π^* the same. This gives a policy π with

$$V(\pi, s, T) = v_j \cdot q_j + q_j(v_i \cdot q_i + q_i \cdot V^*(s \cup \{i,j\}, T-2))$$

- Since π^* is optimal, we know

$$v_i \cdot q_i + q_i(v_j \cdot q_j + q_j \cdot V^*(s \cup \{i,j\}, T-2)) \geq v_j \cdot q_j + q_j(v_i \cdot q_i + q_i \cdot V^*(s \cup \{i,j\}, T-2)).$$

- After re-arranging terms, this implies

$$\frac{v_i \cdot q_i}{1 - q_i} \geq \frac{v_j \cdot q_j}{1 - q_j}$$

Hence, it is optimal to open envelopes in non-increasing order of $v_i q_i/(1-q_i)$. The policy is unique up to tie-breaking among envelopes, for which the fraction has the same value.

Ski-Rental. Reconsider the SKI-RENTAL problem, which we discussed in Chapter 1. Suppose there is a probability $q \in [0,1]$ such that each day the resort opens independently with probability q. We can buy skies at a cost of $B > 1$ or rent at a cost of 1.

SKI-RENTAL as an MDP:

- Three states: (o)pen, (c)losed, boug(h)t skis. Two actions: (r)ent, (b)uy.
- For consistency we use cost instead of reward: $c_a(s) = -r_a(s)$.
- State open: Costs $c_r(o) = 1$, $c_b(o) = B$ and transition: $p_r(o,o) = q$, $p_r(o,c) = 1-q$, $p_b(o,h) = 1$.
- State closed: No costs, and no effect of renting or buying, i.e., $p_r(c,o) = p_b(c,o) = q$ and $p_r(c,c) = p_b(c,c) = 1-q$.
- State bought: No costs, and we remain there $p_b(h,h) = p_r(h,h) = 1$.

- Technically, for MDPs an initial state must be given. But we want that open/closed on first day is random → start from an auxiliary "day 0" in state c.

[Pic: States, Rewards, Transition Probabilities]

The optimal policy π^* is Markovian, depends only on current state and number of days left.

- Under which conditions does it buy in state o? This completely determines the cost.
- $C(T)$ optimal cost for a sequence of T days.
- **Key point**: If π does not buy on day 1, it faces the same MDP with $T-1$ days!
- Hence, from (3.2) we can infer

$$C(T) = \begin{cases} q \cdot (C(T-1)+1) + (1-q) \cdot C(T-1) & \text{if } \pi^* \text{ rents in } (o,T) \\ q \cdot B + (1-q) \cdot C(T-1) & \text{otherwise} \end{cases}$$

- π^* chooses the cheaper of the two options:

$$C(T) = q \cdot \min\{C(T-1)+1, B\} + (1-q) \cdot C(T-1)$$

Analyzing the recursion reveals the optimal buying time:

- We start from $C(0) = 0$. Then $C(T-1)+1 \leq B$ and so $C(T) = C(T-1) + q$.
- Hence, $C(T) = qT$ for the first values of T.
- At some point τ we get $C(\tau-1)+1 > B$. Then it becomes better to buy.
- This point is the smallest τ such that $q(\tau-1)+1 > B$, or $\tau = \lfloor \frac{B-1}{q} + 2 \rfloor$
- Note $C(T-1)+1 > B$ for larger $T \geq \tau$.
- Then $C(T) = qB + (1-q) \cdot C(T-1)$, which approaches B.

Overall, π^* buys on any open day t when there are at least $\lfloor \frac{B-1}{q} + 2 \rfloor$ remaining days (including day t). Clearly, if the first open day satisfies this, it buys on that day. Otherwise, no subsequent open day will satisfy it, so it rents throughout. Note that π^* never first rents and then buys!

Prophet. Consider the PROPHET problem as an MDP.

- Two actions: (a)ccept and (c)ontinue.
- States have to store round t and current value v_t that we decide on
- Action (a) → get reward v_t, move to state (h)alted.
- Action (c) → no reward, move to state $(i+1, r)$ with probability $\Pr[v_{t+1} = r]$.
- In state h, we get no reward and remain in h
- Inititalization: Auxiliary round 0 with state $(0, 0)$
- For simplicity, we assume distributions $\mathcal{D}_t$ are finite → this yields finitely many states.
- In principle the results hold also for general distributions.

[Pic: States, Rewards, Transition Probabilities]

For the optimal policy π^* recall Equation (3.2).

- In each round t only two choices: accept reward v_t, or continue without reward.

- In state (t, v_t), suppose π^* accepts. Then current reward is higher than the optimal expected reward from remaining rounds:

$$v_t \geq \sum_r \Pr[v_{t+1} = r] \cdot V^*((t+1, r), n-t)$$

- Otherwise, if π^* picks continue, the inequality is reversed.
- The right-hand side is the expected reward from an optimal policy in the rounds $t+1, \ldots, n$. Since the policy is Markovian, this is independent of v_t.
- The expected subsequent reward is an *acceptance threshold* for the current round.

This yields the following characterization of the optimal policy π^*.

Theorem 13. *π^* uses thresholds $\tau_1 \geq \ldots \geq \tau_n$ defined recursively by $\tau_n = 0$ and $\tau_t = \mathbb{E}[\max\{v_{t+1}, \tau_{t+1}\}]$ for $1 \leq t < n$. In round t, π^* accepts if $v_t > \tau_t$ and rejects if $v_t < \tau_t$. The decisions are unique up to tie-breaking at $v_t = \tau_t$, which is irrelevant for the expected value.*

Proof. We will show that the sequence τ_t as defined in the theorem is exactly the expected reward of an optimal policy on the subinstance of rounds $t+1, \ldots, n$. That is,

$$\tau_t = \mathbb{E}[V^*((t+1, v_{t+1}), n-t)] \ .$$

Induction on t downward from n to 1.

- Base case $t = n$: trivial. Policy reward is 0, we defined $V^*(s, 0) = 0$ for all states s.
- Step $t \to t-1$: Assume optimal expected reward on $t+1, \ldots, n$ is τ_t
- Recall (3.2). Optimal expected reward on rounds $t, \ldots, n$ for fixed v_t is

$$V^*((t, v_t), n-t+1) = \max_{a \in \mathcal{A}} r_a((t, v_t)) + \sum_{s' \in \mathcal{S}} p_a((t, v_t), s') V^*(s', n-t)$$

- For $a = (a)$ the term is v_t; for $a = (c)$ it is $\mathbb{E}[V^*((t+1, v_{t+1}), n-t)]$
- By hypothesis $\mathbb{E}[V^*((t+1, v_{t+1}), n-t)] = \tau_t$. Thus, for any fixed v_t

$$V^*((t, v_t), n-t+1) = \max(v_t, \tau_t)$$

- Expectation over v_t completes the induction step.

Consider the structure of the thresholds.

- If $v_t > \tau_t$, the maximum is attained for $a = (a)$ccept; if $v_t < \tau_t$, it is attained for $a = (c)$ontinue; if $v_t = \tau_t$, both give same expected reward, so any choice is optimal.
- τ_t is reward of an optimal policy on rounds $t, \ldots, n$. This policy is better than ignoring round t and using the optimal policy for $t+1, \ldots, n$. Hence: $\tau_t \geq \tau_{t+1}$.

□

Example 2. We discuss a small example with $v_1 \sim \text{Uniform}\{0, 1, 2\}$, $v_2 \sim \text{Uniform}\{1, 3, 4, 5\}$, $v_3 \sim \text{Uniform}\{0, 1, 9\}$ and $v_4 \sim \text{Uniform}\{0, 3\}$.

For $\tau_4 = 0$, so $\tau_3 = \mathbb{E}[\max\{v_4, 0\}] = \frac{3}{2}$. For τ_2, we need to compute

$$\tau_2 = \mathbb{E}\left[\max\left\{v_3, \frac{3}{2}\right\}\right] = \frac{2}{3} \cdot \frac{3}{2} + \frac{1}{3} \cdot 9 = \frac{12}{3} = 4$$

Hence, in round 2, the optimal policy has a choice to accept $v_2 = 4$ or continue. For τ_1, we see that

$$\tau_1 = \mathbb{E}\left[\max\left\{v_2, 4\right\}\right] = \frac{3}{4} \cdot 4 + \frac{1}{4} \cdot 5 = 4.25$$

Hence, the expected reward of π^* is $\mathbb{E}[\max\{v_1, \tau_1\}] = \tau_1 = 4.25$. Note that this is the optimal reward of any *online* policy. What is the value of the offline optimum $\mathbb{E}[\max_t v_t]$? ■

3.3 Yao's Principle

In this section, we use distributions and stochastic uncertainty in the input instance more as a analytical tool. Our goal is to lower bound the performance of (not necessarily online) randomized algorithms on worst-case input. Let's consider a cost minimization problem, say, Ski-Rental, and let Σ be a set of possible instances.

First, let's consider input distributions and the best deterministic algorithm:

- There is a random variable S that yields instances from Σ (based on a distribution $\mathcal{D}$).
- Consider the best deterministic (online) algorithm that minimizes the expected cost over the random instances from S.
- Consider the expected cost of that algorithm.

Let's compare this to randomized algorithms and worst-case input:

- Consider any randomized algorithm ALG for the problem, and
- a worst-case instance $\sigma_{ALG} \in \Sigma$ that maximizes the expected cost $\mathbb{E}[ALG(\sigma_{ALG})]$
- Let ALG^* be the algorithm that minimizes $\mathbb{E}[ALG^*(\sigma_{ALG^*})]$

Turns out, the cost in the second case is always larger: For every randomized algorithm ALG there is some worst-case instance σ_{ALG} such that $\mathbb{E}[c(ALG(\sigma_{ALG}))]$ is at least the expected cost of the best deterministic algorithm for any fixed distribution $\mathcal{D}$ over inputs.

Although easy to prove, this insight is very powerful, especially for proving lower bounds. We only have to come up with an *input distribution* $\mathcal{D}$ such that *every deterministic* algorithm performs badly in expectation (over $\mathcal{D}$). Then no *randomized* algorithm can perform well on *worst-case inputs*!

Let's describe the setting a little more formally:

- A randomized algorithm ALG has access to random bits
- If we fix all outcomes of these random bits, ALG becomes deterministic
- Hence: ALG is a **distribution over deterministic algorithms**
- Let $\mathcal{A}$ denote the set of all deterministic algorithms for our problem
- ALG gives rise to a random variable A that takes values from $\mathcal{A}$
- We denote the worst-case expected cost of ALG by $\max_{\sigma \in \Sigma} \mathbb{E}[c(A, \sigma))]$
- In addition, we consider a random variable S to choose instances from Σ.

Yao's principle is per se not about algorithms and instances - it is a general statement on maximization and minimization for functions with random arguments.

Theorem 14 (Yao's Principle)**.** *Let A be an arbitrary random variable taking values in $\mathcal{A}$ and S be a random variable taking values in Σ. Then, for a cost function c we have*

$$\max_{\sigma\in\Sigma} \mathbb{E}[c(A,\sigma)] \quad \geq \quad \min_{a\in\mathcal{A}} \mathbb{E}[c(a,S)] \ .$$

For a value function $v(a,\sigma) = -c(a,\sigma)$ this implies

$$\min_{\sigma\in\Sigma} \mathbb{E}[v(A,\sigma)] \quad \leq \quad \max_{a\in\mathcal{A}} \mathbb{E}[v(a,S)] \ .$$

The two statements are perfectly equivalent. Their specific forms come in handy when lower (upper) bounding the expected cost (value) of randomized algorithms for minimization (maximization) problems, respectively.

Proof. The theorem holds in general, but for simplicity we assume $\mathcal{A}$ and Σ are *finite*. The expectations can be expressed as sums over all outcomes

$$\mathbb{E}[c(A,\sigma)] = \sum_{a\in\mathcal{A}} \Pr[A=a]\cdot c(a,\sigma) \qquad \text{and} \qquad \mathbb{E}[c(a,S)] = \sum_{\sigma\in\Sigma} \Pr[S=\sigma]\cdot c(a,s)$$

- To tackle max and min, we use the fact the maximum entry of a set of numbers is more than any weighted average of that set. Similarly, the minimum entry of a set is less than any weighted average of that set.
- $w_\sigma = \Pr[S=\sigma] \in [0,1]$ can be used as weights for a weighted average, since $\sum_\sigma w_\sigma = 1$.
- Similarly $w_a = \Pr[A=a] \in [0,1]$ can also be used as weights for a weighted average.
- Assembling these insights, and suitably re-arranging the sums we see that

$$\begin{aligned}
\max_{\sigma\in\Sigma} \mathbb{E}[c(A,\sigma)] &= \max_{\sigma\in\Sigma} \sum_{a\in\mathcal{A}} \Pr[A=a]\cdot c(a,\sigma) && \text{by definition}\\
&\geq \sum_{\sigma\in\Sigma} w_\sigma \cdot \sum_{a\in\mathcal{A}} \Pr[A=a]\cdot c(a,\sigma) && \text{since max} \geq \text{weighted average}\\
&= \sum_{\sigma\in\Sigma}\sum_{a\in\mathcal{A}} w_\sigma \cdot \Pr[A=a]\cdot c(a,\sigma) && \text{by rearranging sums}\\
&= \sum_{\sigma\in\Sigma}\sum_{a\in\mathcal{A}} \Pr[S=\sigma]\cdot w_a \cdot c(a,\sigma)\\
&= \sum_{a\in\mathcal{A}} w_a \sum_{\sigma\in\Sigma} \Pr[S=\sigma]\cdot c(a,\sigma) && \text{by rearranging sums}\\
&\geq \min_{a\in\mathcal{A}} \sum_{\sigma\in\Sigma} \Pr[S=\sigma]\cdot c(a,\sigma) && \text{since weighted average} \geq \text{min}\\
&= \min_{a\in\mathcal{A}} \mathbb{E}[c(a,S)] && \text{by definition}
\end{aligned}$$

□

We will apply the principle to show lower bounds on randomized online algorithms that we mentioned earlier.

OnlineMax. We prove Theorem 5 for the OnlineMax problem without random-order or distributional information. We show that every randomized algorithm for OnlineMax has a competitive ratio of $\Omega(n)$.

The key point is to invent a distribution over input instances such that *the best deterministic* algorithm is bad on that input distribution. Towards this end, consider the following stochastic process to generate an input instance.

- The process maintains a condition "alive". Initially, it is alive.
- In each round t, it first updates the alive-status: If it is alive, then it becomes not alive with probability $1/2$. If it is not alive, it stays not alive.
- Then the value for person i is chosen: If alive, $v_t = 2^t$, otherwise $v_t = 0$.

The instances have the form $(2, 4, 8, 16, \ldots, 2^j, 0, \ldots, 0)$. We will call such instances *power-of-2-instances of length* n. The process constructs a random variable S over power-of-2-instances of length n. Round t has non-zero value with probability $(1/2)^t$.

Consider any deterministic online algorithm a for this input distribution.

- In each round t, if it sees $v_t = 0$, it must have value 0.
- Otherwise, it has seen a sequence of powers of 2. It deterministically decides to accept $v_t = 2^t$ or reject it.
- Let i^* be the first round, in which it accepts 2^{t^*}.
- If and only if the process is alive after round t^*, the algorithm gets 2^{t^*}. Otherwise, the algorithm gets value 0. Hence,

$$\mathbb{E}[v(a, S)] = \Pr[\text{alive in round } t^*] \cdot 2^{t^*} = 1$$

Round $t = 1, \ldots, n-1$ is the last alive round with probability $(1/2)^{t+1}$. For round n, this holds with probability $1/2^n$. The offline optimum knows this, it chooses the person from the last alive round. Hence, the expected value of OPT is

$$\mathbb{E}[v(OPT, S)] = \sum_{t=1}^{n} \Pr[t \text{ last alive round}] \cdot 2^t = \frac{n-1}{2} + 1 = \Omega(n)$$

Now, consider any randomized algorithm A. Since it is a distribution over deterministic algorithms, we have

$$\mathbb{E}[v(A, S)] = \sum_{a \in \mathcal{A}} \Pr[A = a] \cdot \mathbb{E}[v(a, S)] = 1$$

Clearly, if A was $o(n)$-competitive, there would be some constant $b \in O(1)$, such that on every power-of-2-instance σ

$$\mathbb{E}[v(A, \sigma)] \geq v(OPT, \sigma)/o(n) + O(1).$$

As such, in expectation over the power-of-2-instances, A would obtain

$$\mathbb{E}[v(A, S)] \geq \Omega(n)/o(n) + O(1) = \omega(1) > 1$$

for sufficiently large n, a contradiction.

3.4 Independent Set

As another important generalization of the OnlineMax we consider classes of the classic Independent Set problem in an online version. Even in the standard offline case, it is NP-hard to approximate this problem within a factor of $n^{1-\varepsilon}$. However, in several meaningful cases, we can show good performance bounds, even with online arrival.

For motivation, consider a facility management agency that schedules events in a location.

- Requests for events arrive online and must be accepted or rejected.
- In month i, we get n requests for events in month $i+1$ sequentially in an online fashion.
- Each event u comes with some start and end date in the next month.
- Event u has to be decided before seeing the next event(s).
- Once accepted, u blocks all other events with an overlap in time.
- **Goal**: Maximize the number of accepted unblocked events.

This is the Interval Scheduling problem discussed in undergraduate classes. Each event is a time interval. The goal is to select as many mutually disjoint intervals as possible. The problem can be framed using an *interval graph* G – each interval corresponds to a node $v \in V$. An edge $\{u, v\}$ exists if and only if the intervals corresponding to u and v overlap. Hence, Interval Scheduling is Independent Set in the interval graph.

We can solve the offline problem optimally with a standard **Greedy Algorithm**:

Accept the unblocked interval with earliest end date; repeat until every interval is either accepted or blocked.

For this algorithm, and throughout this section, we assume that we *break ties consistently*. Hence, w.l.o.g. we assume **all intervals have distinct end dates**.

[Pic: Events, Interval Graph, Greedy Algorithm]

For the online problem, it is impossible to obtain any non-trivial guarantee in the worst-case, even if the number n of intervals is known in advance.

Theorem 15. *Every randomized algorithm for* Online Interval Scheduling *is* $\Omega(n)$*-competitive, even if the number* n *of intervals is known in advance.*

3.4.1 Random Graph and Worst-Case Arrival

Let us instead consider a slightly more structured uncertainty in the Prophet Interval Scheduling problem:

- There is a known base graph $G = (V, E)$ with n nodes. G is an interval graph.
- G is a *supergraph* of the actual (random) input graph.
- The n intervals of V are considered in an arbitrary (i.e., possibly worst-case) order
- In the beginning, each interval v has a known probability p_v to exist.
- Round t: Nature decides if interval u_t exists by independent draw with prob. p_{u_t}

Consider Algorithm 6 for this problem. The algorithm needs to know for each node an **interval representation** of start and end date. Looking more closely, it is indeed sufficient to know **only the ordering** of all n intervals by end date.

Algorithm 6: Prophet Interval Scheduling

1 $V^S \leftarrow \emptyset$ `// sample set of intervals`
2 **for** *each node* $v \in V$ **do** with probability p_v add v to V^S
3 $M_1 =$ Independent Set chosen by Greedy Algorithm on the induced subgraph $G[V^S]$
4 $M_2, M_3, M_4 \leftarrow \emptyset$ `// filtered sets of input intervals`
5 **for** *rounds* $t = 1, \ldots, n$ **do**
6 **if** u_t *does not exist* **then** continue to next round
7 **else if** $\nexists\, v \in M_1$ *with earlier end date and* $\{v, u_t\} \in E$ **then** add u_t to M_2
8 **if** $u_t \in M_2$ **then** with prob. 1/2: add u_t to M_3
9 **if** $u_t \in M_3$ *and* u_t *has no edges to nodes in* M_4 **then** add u_t to M_4
10 **return** M_4

Theorem 16. *Algorithm 6 is* 4*-competitive for the* Prophet Interval Scheduling *problem.*

Let V^I be the (random) set of nodes that actually exist in the input graph. The (random) input graph is $G[V^I]$, the subgraph of G induced by V^I. Let $OPT(V^I)$ be the size of the largest independent set in $G[V^I]$.

Lemma 5. *It holds that* $\mathbb{E}[|M_1|] = \mathbb{E}[OPT(V^I)]$.

Proof. Not knowing V^I, the algorithm instead draws the sample V^S. M_1 computed by Greedy is an optimal solution on $G[V^S]$. Since V^I and V^S are drawn independently from the same distribution

$$\mathbb{E}[|M_1|] = \mathbb{E}[OPT(V^S)] = \mathbb{E}[OPT(V^I)].$$

□

Clearly, the algorithm discards all nodes v_t that do not exist. Among the ones that exists, it should accept unblocked intervals that block few subsequent ones. It does so using two filtering steps in lines 7 and 8. Maybe surprisingly, the first step has no impact on the expected size of the optimal solution.

Let X_u^i be an indicator variable that $u \in M_i$, for any node $u \in V$ and $i = 1, 2, 3$.

Lemma 6. *For every interval* u*, it holds that*

$$\mathbb{E}[X_u^1] = \mathbb{E}[X_u^2].$$

Furthermore, for every overlapping interval v *with earlier end date, it holds*

$$\mathbb{E}[X_u^1 \mid v \in M_3] = \mathbb{E}[X_u^2 \mid v \in M_3].$$

Proof. For the first statement, look at the round t with $u_t = u$.

- First, suppose there is some overlapping interval $v \in M_1$ with earlier end date than u. Then $X_u^2 = 0$. Also, $X_u^1 = 0$, since Greedy would consider u later and reject it.

- Second, suppose there is no overlapping interval $v \in M_1$ with earlier end date than u.
- Greedy will include u in M_1 if $u \in V^S$. Similarly, the algorithm adds it to M_2 if $u \in V^I$. Hence,

(a) $u \in V^S$, $u \notin V^I \Rightarrow X_u^1 = 1$, $X_u^2 = 0$.
(b) $u \notin V^S$, $u \in V^S \Rightarrow X_u^1 = 0$, $X_u^2 = 1$.
(c) $u \in V^S$, $u \in V^I \Rightarrow X_u^1 = X_u^2 = 1$.
(d) $u \notin V^S$, $u \notin V^I \Rightarrow X_u^1 = X_u^2 = 0$

- M_1 and M_2 treat u differently only in cases (a) and (b). They have the same probability since sample is independent of input: $\Pr[u \in V^S, u \notin V^I] = \Pr[u \notin V^S, u \in V^I]$.

This proves $\mathbb{E}[X_u^1] = \mathbb{E}[X_u^2]$.

The second statement follows very similarly. Suppose for u there is some overlapping $v \in M_3$ with earlier end date.

- First, consider event A that M_1 contains v or any other overlapping interval with earlier end date than u. Then

$$\mathbb{E}[X_u^1 \mid A, v \in M_3] = \mathbb{E}[X_u^2 \mid A, v \in M_3] = 0.$$

- Second, assume otherwise. Then if $u \in V^S$, it is added to M_1 by Greedy. If $u \in V^I$, it is added to M_2 in line 7. $v \in M_3$ has no influence on this decision. Hence,

$$\mathbb{E}[X_u^1 \mid \neg A, v \in M_3] = \Pr[u \in V^S] = p_u = \Pr[u \in V^I] = \mathbb{E}[X_u^2 \mid \neg A, v \in M_3].$$

□

As a consequence of the first statement, $\mathbb{E}[|M_2|] = \mathbb{E}[|M_1|]$. Also, clearly, $\mathbb{E}[|M_3|] = \mathbb{E}[|M_2|]/2$ due to line 8. Now the following lemma implies a relation between M_3 and M_4. Consider the set of all edges among nodes in M_3

$$C = \{\{u, v\} \in E \mid u, v \in M_3\}.$$

Lemma 7. *It holds that* $\mathbb{E}[|C|] \leq \mathbb{E}[|M_3|]/2$.

Proof. For any fixed node $v \in M_3$. Consider all intervals with *later* end date that would be blocked by v

$$C_v = \{u \in V \mid \{u, v\} \in E \text{ and } u \text{ has later end date }\}.$$

[Pic: Set C_v of intervals that overlap with $v \in M_3$ and have later end date]

Suppose v would enter M_4.

- How many of the intervals in C_v would actually be rejected because $v \in M_4$?
- To be blocked by v, an interval from C_v must first make it to M_3 in line 8! Note that

$$\mathbb{E}[|C_v \cap M_3| \mid v \in M_3] = \mathbb{E}\left[\sum_{u \in C_v} X_u^3 \,\middle|\, v \in M_3\right] = \sum_{u \in C_v} \mathbb{E}\left[X_u^3 \mid v \in M_3\right].$$

- Clearly, $\mathbb{E}[X_u^3 \mid v \in M_3] = \mathbb{E}[X_u^2 \mid v \in M_3]/2$ due to line 8.

- The second statement of the previous lemma implies $\mathbb{E}[X_u^2 \mid v \in M_3] = \mathbb{E}[X_u^1 \mid v \in M_3]$, and hence
$$\sum_{u \in C_v} \mathbb{E}\left[X_u^3 \mid v \in M_3\right] = \frac{1}{2} \sum_{u \in C_v} \mathbb{E}\left[X_u^1 \mid v \in M_3\right]$$
- By construction, all intervals in C_v are mutually overlapping. Hence, there can be at most one interval from C_v in M_1.
$$\sum_{u \in C_v} \mathbb{E}\left[X_u^1 \mid v \in M_3\right] \leq 1.$$
- Hence, in expectation v blocks at most 0.5 intervals from C_v!

Now consider the set C. Clearly, each edge among nodes in C can be assigned to the node v with earlier end date, so $|C| = \sum_{v \in M_3} |C_v \cap M_3|$

$$\begin{aligned}
\mathbb{E}[|C|] &= \sum_{v \in C} \Pr[v \in M_3] \cdot \mathbb{E}[|C_v \cap M_3| \mid v \in M_3] \\
&= \sum_{v \in C} \Pr[v \in M_3] \cdot \sum_{u \in C_v} \mathbb{E}[X_u^3 \mid v \in M_3] \\
&= \sum_{v \in C} \Pr[v \in M_3] \cdot \frac{1}{2} \sum_{u \in C_v} \mathbb{E}[X_u^1 \mid v \in M_3] \\
&\leq \frac{1}{2} \sum_{v \in C} \Pr[v \in M_3] \cdot 1 \quad = \quad \mathbb{E}[|M_3|]/2
\end{aligned}$$

□

Proof of Theorem 16. C contains at least one unique edge for every node from M_3 that is denied to enter M_4. Hence $|M_4| \geq |M_3| - |C|$. Overall, the expected number of intervals accepted by Algorithm 6 is

$$\begin{aligned}
\mathbb{E}[|M_4|] &\geq \mathbb{E}[|M_3| - |C|] \\
&\geq 1/2 \cdot \mathbb{E}[|M_3|] \\
&= 1/4 \cdot \mathbb{E}[|M_2|] = 1/4 \cdot \mathbb{E}[|M_1|] \\
&= 1/4 \cdot \mathbb{E}[OPT(V^I)]
\end{aligned}$$

□

3.4.2 Worst-Case Graph and Random-Order Arrival

Algorithm 6 can be applied beyond the rather special uncertainty model of Prophet Interval Scheduling. It can be adapted to many other mixtures of stochastic and worst-case uncertainty. Let us discuss this for a Secretary Interval Scheduling problem:

- Base graph G is **unknown** (e.g., determined by an adversary) in advance
- **Properties of G known** in advance: G is interval graph, number n of nodes

Algorithm 7: Secretary Interval Scheduling

$V^S \leftarrow \emptyset$ // sample set of intervals
$k \sim$Binom$(n, 1/2)$ // random number of sample rounds
for *rounds* $t = 1, \ldots, k$ **do** add u_t to V^S
$M_1 =$ Independent Set chosen by Greedy Algorithm on the induced subgraph $G[V^S]$
$M_2, M_3, M_4 \leftarrow \emptyset$
for *rounds* $t = k+1, \ldots, n$ **do**
  if $\nexists\, v \in M_1$ *with earlier end date and* $\{v, u_t\} \in E$ **then** add u_t to M_2
  if $u_t \in M_2$ **then** with prob. 1/2: add u_t to M_3
  if $u_t \in M_3$ *and* u_t *has no edges to nodes in* M_4 **then** add u_t to M_4
return M_4

- After construction of G, the n nodes arrive sequentially in **uniform random order**
- Upon arrival in round t, node u_t reveals all edges to nodes arrived in earlier rounds
- u_t must be accepted or rejected before seeing the next node(s), decisions are irrevocable
- Goal: Construct a large independent set.

We solve this problem with a variant of Algorithm 6 given in Algorithm 7.
Some remarks:

- We build a sample V^S as the set of nodes in the first k rounds.
- Define the "input" set of nodes V^I as the ones arriving in rounds $k+1, \ldots, n$.
- Note that $V^S \cap V^I = \emptyset$, i.e., a node u is either in the sample or in the input.
- We assume to get an interval representation for every arriving node. As before, an ordering of all (arrived so far) nodes w.r.t. non-decreasing end date is sufficient.
- W.l.o.g. we can break ties arbitrarily and assume that all end dates are distinct, i.e., the ordering is unique.

Theorem 17. *Algorithm 7 is 8-competitive for the* Secretary Interval Scheduling *problem.*

To show the theorem, we largely adapt the proof from the previous section.

We start by observing that the sampling procedure has an independence property. Given any base graph $G = (V, E)$ and a node $u \in V$, the events $u \in V^S$ and $u \in V^I$ are mutually exclusive, and exactly one of the events occurs. We observe that the events are independent for different nodes $u \neq v$.

Lemma 8. *Consider any two nodes* $u \neq v$ *from* V*. The events* $u \in V^x$ *and* $v \in V^y$ *are independent, for every* $x, y \in \{I, S\}$*.*

Proof. Nature picks the permutation π of arrival uniformly at random. The algorithm picks a sample length k by flipping n independent coins with probability 1/2 and counting the number of heads (i.e., draws from a binomial distribution with parameters n and $p = 1/2$). These two stochastic processes are independent.

Consider the following **simulation**:

1. For each $v \in V$, flip independent coin with prob. 1/2 whether $v \in V^S$ or V^I.
2. Place sample intervals in the front of π, input intervals in the back of π
3. Permute sample in uniform random order in the front of π.
4. Permute input in uniform random order in the back of π.

[Pic: Simulation vs. original process]

Obviously, the lemma holds for the simulation (due to step 1.). We now show that the simulation yields an equivalent outcome as the original processes. This follows from two properties:

(1) Given any sample length $k \in \{0, 1, \ldots, n\}$, the arrival order of V in the simulation is uniform over all permutations.
(2) Given any arrival order, the sample length in the simulation is binomially distributed with parameters n and $p = 1/2$.

We leave the proof of these properties as an exercise. □

Note that in Secretary Interval Scheduling the interval graph G is not subject to randomization (only the arrival order). As such, the size OPT of the maximum independent set is not a random variable.

Lemma 9. *It holds that* $\mathbb{E}[|M_1|] \geq OPT/2$.

Proof. Consider M^*, an optimal independent set in G with $|M^*| = OPT$. By Lemma 8 each node from M^* is in V^S independently with probability 1/2. Hence, using indicator variables for $v \in V^S$ and linearity of expectation

$$\mathbb{E}[|M^* \cap V^S|] = \sum_{v \in M^*} \Pr[v \in V^S] = |M^*|/2 = OPT/2.$$

Clearly, since M_1 is the optimum for V^S, we know $|M_1| \geq |M^* \cap V^S|$ for every realization of the sample set V^S. The lemma is proved. □

We again use X_u^i as an indicator variable that $u \in M_i$, for any node $u \in V$ and $i = 1, 2, 3$.

Lemma 10. *For every interval u, it holds that*

$$\mathbb{E}[X_u^1] = \mathbb{E}[X_u^2].$$

Furthermore, for every overlapping interval v with earlier end date, it holds

$$\mathbb{E}[X_u^1 \mid v \in M_3] = \mathbb{E}[X_u^2 \mid v \in M_3].$$

Proof. The proof of this lemma is almost the same as the proof of Lemma 6 above.

In the proof of the first statement, cases (c) and (d) do not occur. Cases (a) and (b) have the same probability since $\Pr[u \in V^S] = \Pr[u \in V^I] = 1/2$.

For the second statement, $v \in M_3$ again has no influence on the decision if $u \in M_1$ or $u \in M_2$. Considering event A that M_1 contains an overlapping interval with earlier end date than u (Note: this cannot be v here, since $v \in M_3$ implies $v \in V^I$ and $v \notin V^S$), then conditioned on A and $\neg A$ both events $u \in M_1$ and $u \in M_2$ have the same probabilities, respectively. □

The first statement implies $\mathbb{E}[|M_2|] = \mathbb{E}[|M_1|]$. Also, clearly, $\mathbb{E}[|M_3|] = \mathbb{E}[|M_2|]/2$. For the relation between M_3 and M_4 we again consider the set $C = \{\{u, v\} \in E \mid u, v \in M_3\}$ of all edges among nodes in M_3.

Lemma 11. *It holds that* $\mathbb{E}[|C|] \leq \mathbb{E}[|M_3|]/2$.

Lemma 10 shows the same properties as Lemma 6, and the structure of the algorithm with construction of M_2, M_3 and M_4 is exactly the same as before. The proof of Lemma 7 can be applied directly without modifications.

Proof of Theorem 17. We combine the properties exactly as in the proof of Theorem 16, using the additional factor of 2 relating $\mathbb{E}[|M_1|]$ and OPT:

$$\begin{aligned}\mathbb{E}[|M_4|] &\geq \mathbb{E}[|M_3| - |C|] \\ &\geq 1/2 \cdot \mathbb{E}[|M_3|] \\ &= 1/4 \cdot \mathbb{E}[|M_2|] = 1/4 \cdot \mathbb{E}[|M_1|] \\ &= 1/8 \cdot OPT\end{aligned}$$

□

3.4.3 Inductive Independence and Graph Sampling

We have seen two scenarios in which the same algorithmic idea can be applied to obtain a constant competitive ratio. How far can this idea and the proof arguments be extended? What are key properties of the uncertainty model that make the arguments work?

As an important extension, we consider the structure of the base graph G. The main idea in the algorithms above is **not restricted to interval graphs**! The key property is that there is (and we know) a *good ordering* π of the nodes:

- For ordering π and each node $v \in V$ consider the *backwards neighborhood*

$$N^+_{v,\pi} = \{u \in V \mid \{u, v\} \in E, u \succ_\pi v\},$$

  i.e., all neighbors of v that come later in ordering π.
- Consider the size of the largest independent set in the backwards neighborhood

$$\rho_{v,\pi} = \max\{|I| \mid I \subseteq N^+_{v,\pi} \text{ and } I \text{ is independent set }\}.$$

- Now for ρ_π take the largest size for all nodes,

$$\rho_\pi = \max_{v \in V}\{\rho_{v,\pi}\}.$$

  For ordering π every independet set in every backwards neighborhood is at most ρ_π.
- A good ordering π gives a small ρ_π. The best ordering yields the **inductive independence number** ρ_G of G:

$$\rho_G = \min_\pi\{\rho_\pi\}$$

A bounded inductive independence number implies that a good independent set can be computed easily. The proof of the following proposition is left as an exercise.

Proposition 1. *Given any graph G and an ordering π of the nodes, consider the standard greedy algorithm that considers all nodes in the order of π and adds them to the indepedent set whenever possible. The algorithm computes a ρ_π-approximation.*

In the next propsition, we consider non-trivial interval graphs G with $E \neq \emptyset$.

Proposition 2. *Every non-trivial interval graph has $\rho_G = 1$. The best ordering is the one by non-increasing end date.*

Proof. Consider the ordering π by non-decreasing end dates.

- $N^+_{u,\pi}$ is the set of all intervals with later end date that overlap u.
- All intervals in $N^+_{u,\pi}$ must overlap each other, since they cross end date of u.
- Hence, the largest independent set in $N^+_{u,\pi}$ has size 1 for every node u.
- $\rho_\pi = 1$ for the ordering π by non-decreasing end dates.
- Note: $\rho_\pi \geq 1$ for every ordering π unless $E = \emptyset$.

Hence, for every interval graph with $E \neq \emptyset$ we have $\rho_G = 1$. □

$\rho_G = 1$ implies that G is a **chordal graph** and the best ordering π is an *elimination order*. In fact, an elimination order for a chordal graph can be computed in polynomial time (just from the graph structure, without interval representations).

In many interesting classes of geometric intersection graphs (such as disk graphs, or other conflict graphs derived from interference models in wireless networks) the geometric interpretation suggests an ordering π, for which ρ_π is known to be small.

[Pic: Disk graphs, bounded inductive independence]

Algorithms 6 and 7 can be applied rather directly to such graphs:

- For the arriving graph they need to know (a) an ordering π and (b) the number ρ_π.
- This must be known for the sample $G[V^S]$ and in the induced subgraph of sample and arrived nodes in every round t.
- The algorithms get adjusted in the following way:
  (a) Use the Greedy algorithm from Proposition 1 based on π to compute M_1.
  (b) Change the probability of $1/2$ to $1/(2\rho_\pi)$ when adding a node $u_t \in M_2$ to M_3.
- Each of these adjustments deteriorates the ratio by a factor of ρ_π.

Theorem 18. *Given any base graph G with an ordering π, there are algorithms that are $4\rho_\pi^2$-competitive for* Prophet Independent Set *and $8\rho_\pi^2$-competitive* Secretary Independent Set.

The proof of the theorem is left as an exercise.

Beyond the structure of the base graph G, we also generalize the stochastic subgraph generation and arrival model. Consider a **graph sampling model** with the following properties:

- The base graph $G = (V, E)$ is unknown, $n = |V|$ is known. G has an ordering π with small inductive independence number.

- In the beginning, we see a node sample $V^S \subseteq V$, the induced subgraph $G[V^S]$, and the ordering of these nodes according to π.
- Then the input nodes $V^I \subseteq V$ arrive in worst-case order. Each arriving node reveals all incident edges to nodes in V^S and previously arrived ones from V^I. We see the ordering of all sample and arrived input nodes according to π.
- The goal is to compute a good independent set in $G[V^I]$.
- V^S and V^I are generated stochastically from G according to two properties:
- **Stochastic Similarity**: For each $v \in V$, the probabilities to be in V^S and V^I are similar up to a factor c, i.e., $\Pr[v \in V^S] \leq c \cdot \Pr[v \in V^I]$ and $\Pr[v \in V^I] \leq c \cdot \Pr[v \in V^S]$ for some $c \geq 1$.
- **Independence Across Nodes**: For $u \neq v$ the events $u \in V^x$ and $v \in V^y$ are independent, for all $x, y \in \{I, S\}$. For the same node v, events $v \in V^S$ and $v \in V^I$ can be arbitrarily correlated!

In the graph sampling model, we stochastically draw sample and input graphs from the same underlying graph G. Sample and input node sets are related by a factor c. For different nodes, the events that the nodes show up in sample and/or input are independent. Other properties, such as correlation of being in sample and input for the same node, arrival order of the input set, or the graph structure of G, are not crucial and can be chosen in a worst-case fastion by an adversary.

Both prophet and secretary variants can be seen as special cases of the graph sampling model. The main extension is the stochastic similarity property – both variants above satisfied this property with $c = 1$. The algorithms and the proof structure can be extended rather directly.

Corollary 1. *There is an algorithm that is $4\rho_\pi^2 c^3$-competitive for* Online Independent Set *in the graph sampling model.*

Chapter 4

Probing and Testing

4.1 k-Probing and Adaptivity Gap

We consider the k-ProbeMax problem:

- There are n closed boxes. Each box i contains some prize of non-negative value v_i.
- Each box i has a distribution $\mathcal{D}_i$ written on it. The prize inside is drawn $v_i \sim \mathcal{D}_i$ independently at random.
- $k \leq n$ boxes can be opened. If box i is opened, the realization of v_i is revealed.
- After k boxes are opened, we can choose one opened box. We receive the prize of the chosen box.
- **Goal**: Maximize the value of the prize in the chosen box

[Pic: Boxes, Prizes, Probes]

In the k-ProbeMax problem there is no online arrival with immediate and irrevocable decisions. Instead, we can **probe** (i.e., open the box and see the realized value of) a number of choices. Which boxes should be opened such that we maximize the expected value of the prize in the end?

k-ProbeMax is easily expressed as a Markov decision process:

- State: Subset S of opened boxes and maximum reward v_S^* of any opened box in S.
- Action: Unopened box ("reopening" a box $\rightarrow$ infinitely negative reward).
- Open box a in state (S, v_S^*): Reward $r_a(S, v_S^*)$ is the expected increase in highest prize:

$$r_a(S, v_S^*) = \mathbb{E}[\max(v_a - v_S^*, 0)]$$

- Transition probabilities: $p_a((S, v_S^*), (S \cup \{a\}, v_S^*)) = \Pr[v_a \leq v_S^*]$ and $p_a((S, v_S^*), (S \cup \{a\}, x) = \Pr[v_a = x]$ for all $x > v_S^*$.

Example 3. Consider $n = 3$ boxes and $k = 2$ probes. The distributions are

- Box 1 has $v_1 = 40$ with probability $1/4$, and 0 otherwise.
- Box 2 has $v_2 = 32$ with probability $1/2$, and 0 otherwise.
- Box 3 has $v_3 = 16$ with probability 1.

Suppose our policy fixes the two boxes to open in advance. The expected prize values are

Open 1 and 2:

$$\mathbb{E}[\max\{v_1, v_2\}] = \frac{1}{4} \cdot 40 + \frac{3}{4} \cdot \frac{1}{2} \cdot 32 = 22$$

Open 1 and 3:

$$\mathbb{E}[\max\{v_1, v_3\}] = \frac{1}{4} \cdot 40 + \frac{3}{4} \cdot 16 = 22$$

Open 2 and 3:

$$\mathbb{E}[\max\{v_2, v_3\}] = \frac{1}{2} \cdot 32 + \frac{1}{2} \cdot 16 = 24$$

Hence, the best of these policies obtains a value of 24. It turns out that the optimal policy can do better. The policies above are **non-adaptive** in the sense that they do not adapt to the revealed content of the boxes. What if we choose boxes **adaptively**?

Suppose we first open box 2. With probability 1/2 it has value 32. Then it makes no sense opening the inferior box 3 – we explore whether we can improve upon 32 and open box 1. Otherwise, with probability 1/2 box 2 has value 0. Then it is better to choose box 3, since the expected value is $16 > 40/4 = 10$. Overall, the value of this adaptive strategy is

$$\frac{1}{2} \cdot \left(\frac{1}{4} \cdot 40 + \frac{3}{4} \cdot 32 \right) + \frac{1}{2} \cdot 16 = 17 + 8 = 25$$

■

Adaptive policies can be highly complicated. An adaptive policy yields a decision tree that gives for each outcome of the probes the next box to pick. This tree can have an exponential number of leaves, which can makes it hard to compute and extremely large to represent.

Non-adaptive policies for MDPs are much easier to handle – they choose a fixed sequence of actions in advance, irrespective of the state transitions and rewards seen throughout the process. An interesting question is to bound the **impact of the restriction to non-adaptive policies** on the expected reward.

For a Markov decision process, the **adaptivity gap** is given by the ratio of expected rewards between the best non-adaptive policy and the optimal (adaptive) policy π^*

$$\max_{\pi \text{ non-adaptive}} \frac{V(\pi^*, s_{init}, T)}{V(\pi, s_{init}, T)}$$

The example shows that the adaptivity gap is at least $25/24 = 1.041\bar{6}$. The best upper bound that is currently known is the following.

Theorem 19. *The adaptivity gap for k-*ProbeMax *is at most* $\mathsf{e}/(\mathsf{e}-1) \approx 1.58$.

We only prove an easier bound of 8, i.e., we construct a non-adaptive policy and show that it achieves at least 1/8 of the expected reward of π^*. The better bound in the theorem is shown using an approach that extends the one we discuss here.

The idea is to derive a linear program (LP) and solve it. The LP is constructed such that the optimal value is more than the expected reward of the optimal adaptive policy π^*. We then

transform the optimal LP-solution into a non-adaptive policy. The transformation incurs only a loss of a factor of 1/8 in value and proves the bound.

Fix the optimal policy π^* and consider an execution. We define the following indicator variables along with some notation:

- $X_i = 1$ if box i is opened, and 0 otherwise.
- $Y_{i,v} = 1$ if box i contains prize v and is chosen, and 0 otherwise.
- $x_i = \mathbb{E}[X_i] = \Pr[i \text{ opened}]$
- $y_{i,v} = \mathbb{E}[Y_{i,v}] = \Pr[v_i = v \text{ and } i \text{ is (opened and) chosen}]$
- $q_{i,v} = \Pr[v_i = v]$

Example 3 (continued)**.** We apply the definitions to the optimal policy in our example. Box 2 is always opened ($x_2 = 1$), boxes 1 and 3 are opened only with prob. 1/2 ($x_1 = x_3 = 1/2$). For the values, we consider the combination of realization and selection. If box 2 has value 30, we open box 1. We choose it only when it has value 36. This combination of events happens with probability $y_{1,40} = (1/2) \cdot (1/4) = 1/8$. If box 1 has value 0, we pick box 2. This combination of events happens with probability $y_{2,32} = (1/2) \cdot (3/4) = 1/8$. If box 2 has value 0, we select box 3, since it has always value 16. This event happens with probability $y_{3,16} = 1/2$. The remaining values are $y_{i,0} = 0$, for $i = 1, 2$. ■

We express the expected reward of π^* using x_i and $y_{i,v}$, and we derive further necessary conditions for these values.

- By linearity of expectation and the definitions above

$$V(\pi^*, s_{init}, T) = \mathbb{E}\left[\sum_{i,v} v \cdot Y_{i,v}\right] = \sum_{i,v} v \cdot \mathbb{E}[Y_{i,v}] = \sum_{i,v} v \cdot y_{i,v}.$$

- π^* can open at most k boxes, i.e., $\sum_i X_i \leq k$ with probability 1. Clearly, this implies

$$\sum_i x_i = \sum_i \mathbb{E}[X_i] = \mathbb{E}\left[\sum_i X_i\right] \leq k.$$

- π^* can choose at most one box, i.e., $\sum_{i,v} Y_{i,v} \leq 1$ with probability 1. Hence

$$\sum_{i,v} y_{i,v} = \sum_{i,v} \mathbb{E}[Y_{i,v}] = \mathbb{E}\left[\sum_{i,v} Y_{i,v}\right] \leq 1.$$

- $Y_{i,v} = 1$ if box i is chosen, contains prize v, and v is the maximal prize in any opened box. We drop the last condition and see

$$y_{i,v} = \Pr[Y_{i,v} = 1] \leq \Pr[X_i = 1 \text{ and } v_i = v] = \Pr[X_i = 1] \cdot \Pr[v_i = v]$$

  The latter equality holds since opening a box is a decision of π^*, but the value v_i is realized according to an independent draw from $\mathcal{D}_i$. Note that

$$y_{i,v} \leq \Pr[X_i = 1] \cdot \Pr[v_i = v] = x_i \cdot q_{i,v} \tag{4.1}$$

Algorithm 8: Non-Adaptive Policy for k-ProbeMax

Solve the LP, let (x^*, y^*) be an optimal solution
for $i = 1, \ldots, n$, *until* k *boxes are open* **do** open box i with probability $x_i^*/4$
Inspect all opened boxes, pick i^* as opened box with best prize
return i^* and v_{i^*}

Algorithm 9: Simulation

Solve the LP, let (x^*, y^*) be an optimal solution
for $i = 1, \ldots, n$, *until* k *boxes are open* **do**
    open box i with probability $x_i^*/4$
    if i *open* **then**
        Let v be the value of box i
        With prob. $y_{i,v}^*/(x_i^* \cdot q_{i,v})$: break from loop `// is in` $[0,1]$ `because` (4.1)
return last opened box and its value

The above constraints are linear in the variables x_i and $y_{i,v}$. We combine the objective function and the constraints into the following LP:

$$\begin{aligned} \text{Maximize} \quad & \sum_{i,v} v \cdot y_{i,v} \\ \text{subject to} \quad & \sum_{i} x_i \leq k \\ & \sum_{i,v} y_{i,v} \leq 1 \\ & y_{i,v} \leq x_i \cdot q_{i,v} \quad \forall i, v \\ & x_i, y_{i,v} \in [0,1] \quad \forall i, v \end{aligned} \tag{4.2}$$

Setting all x_i and $y_{i,v}$ according to an optimal policy π^* gives *some feasible solution* to the LP, and its value is the expected reward of π^*. An *optimal LP-solution* only has larger value.

In fact, the optimal LP solution can have a *strictly* larger value than the reward of π^*. Hence, there are LP solutions that do not correpsond to probabilities in policies: Consider $n = 2$ boxes and $k = 2$. Each box has value 1 with probability 1/2, and 0 otherwise. The optimal policy opens both boxes ($x_1 = x_2 = 1$). For the selection w.l.o.g. break ties in favor of box 1: $y_{1,1} = 1/2$ (box 1 has value 1), $y_{2,1} = 1/4$ (box 1 has value 0, box 2 value 1) and $y_{1,0} = 1/4$ (both boxes value 0). The reward is 3/4. In contrast, the optimal LP solution sets $x_1 = x_2 = 1$, and $y_{1,2} = y_{2,1} = 1/2$ and gets a value of 1.

We use Algorithm 8 to compute a non-adaptive policy for opening the boxes.

Theorem 20. *Algorithm 8 obtains an expected reward of at least* $\frac{1}{8}\sum_{i,v} v \cdot y_{i,v}^*$. *Hence, the adaptivity gap for* k-ProbeMax *is at most 8.*

Proof. Consider the simulation in Algorithm 9. The simluation clearly yields inferior reward, since it replaces choosing the best opened box in line 3 by a different, probabilistic choice. Upon opening a box, the algorithm directly decides to choose the box (by breaking from the loop) or discard it. Hence, the simulation is solving a harder, online version of the problem in the spirit of the PROPHET problem.

We show the theorem for the inferior expected reward achieved by the simulation.

- Again, use indicator variable $Y_{i,v} = 1$ if box i is opened, contains value v and is chosen, (and $Y_{i,v} = 0$ otherwise)
- Loop does not reach box $i \Rightarrow Y_{i,v} = 0$.
- Events (a) Reach box i, (b) Open box i, (c) Choose box i are independent:
- (a) depends only on coin flips in rounds $1, \ldots, i$,
- (b) depends on independent coin flip to decide the opening,
- (c) depends on independent coin flip based on value v of box i, which is irrelevant for other events

Hence,

$$\begin{aligned}\Pr[Y_{i,v} = 1] &= \Pr[Y_{i,v} = 1 \mid \text{loop reaches box } i] \cdot \Pr[\text{loop reaches box } i] \\ &= \frac{x^*_i}{4} \cdot q_{i,v} \cdot \frac{y^*_{i,v}}{x^*_i \cdot q_{i,v}} \cdot \Pr[\text{loop reaches box } i] \\ &= \frac{y^*_{i,v}}{4} \cdot \Pr[\text{loop reaches box } i]\end{aligned}$$

We show below that $\Pr[\text{loop reaches box } i] \geq 1/2$. Then the expected reward is

$$\mathbb{E}\left[\sum_{i,v} Y_{i,v} \cdot v\right] = \sum_{i,v} v \cdot \Pr[Y_{i,v} = 1] \geq \sum_{i,v} v \cdot \frac{y^*_{i,v}}{4} \cdot \frac{1}{2} = \frac{1}{8} \cdot \sum_{i,v} v \cdot y^*_{i,v}$$

and the theorem is proved.

Now $\Pr[\text{loop reaches box } i] \geq 1/2$ follows when $\Pr[\text{loop does not reach box } i] \leq 1/2$. Using a union bound, we see

$$\begin{aligned}&\Pr[\text{loop does not reach box } i] \\ &= \Pr[\text{at least } k \text{ boxes opened or at least one box chosen in rounds } 1, \ldots, i-1] \\ &\leq \Pr[\text{at least } k \text{ boxes opened in rounds } 1, \ldots, i-1] \\ &\quad + \Pr[\text{at least one box chosen in rounds } 1, \ldots, i-1]\end{aligned}$$

For each of the latter probabilities, we use Markov inquality to bound them to at most $1/4$.

Suppose $X_i = 1$ if box i is opened, and 0 otherwise. For the expected number of boxes opened in rounds $1, \ldots, i-1$

$$\mathbb{E}\left[\sum_{i'<i} X_{i'}\right] = \sum_{i'<i} \Pr[X_{i'} = 1] \leq \sum_{i'<i} \Pr[X_{i'} = 1 \mid \text{loop reaches box } i'] = \sum_{i'<i} x^*_{i'}/4 \leq \frac{k}{4}$$

since (x^*, y^*) is a feasible solution to LP (4.2). Hence, by Markov inequality,

$$\Pr[\text{at least } k \text{ boxes opened in rounds } 1, \dots, i-1] \leq 1/4.$$

Similarly, for the expected number of boxes chosen in rounds $1, \dots, i-1$

$$\mathbb{E}\left[\sum_{i'<i,v} Y_{i',v}\right] = \sum_{i'<i,v} \Pr[Y_{i',v} = 1] \leq \sum_{i'<i,v} \Pr[Y_{i',v} = 1 \mid \text{loop reaches box } i'] = \sum_{i'<i,v} y^*_{i',v}/4 \leq \frac{1}{4}$$

and, by Markov inequality,

$$\Pr[\text{at least one box chosen in rounds } 1, \dots, i-1] \leq 1/4.$$

□

4.2 k-Testing

Rather than probing a decision alternative and learning its precise value, in many applications we can only *test* it, and the test gives some partial information about the value. Consider, for instance, n different vaccines for a disease. We only learn the effectiveness of a vaccine via testing, where repeated testing leads to a more precise estimate about the exact performance. Likewise, consider hiring a job candidate. As part of the application process, we invite candidates for an interview and test their abilities. Repeated testing of a candidate leads to a more precise estimate on the quality of that candidate.

There are many plausible ways how to model a test that partially reveals the valuations of an alternative. For simplicity we consider the following basic k-TestMax problem:

- There are n closed boxes. Each box i contains some prize of non-negative value v_i.
- Each box i has a distribution $\mathcal{D}_i$ written on it. The realization of the prize inside is drawn initially $v_i \sim \mathcal{D}_i$ independently at random.
- There are k tests. Each test can be applied to any of the boxes.
- If box i gets tested, we get a **new distribution $\mathcal{D}_i$ that describes the realization of box i more precisely**.
- After k tests, choose any box that was tested at least once. We get the realization of the prize in the chosen box.
- **Goal**: Maximize the prize in the chosen box

What happens when we **test a box**?

- In the beginning, nature draws realization of box i, let the value be $v'_i \sim \mathcal{D}_i$.
- We define regions of the support of $\mathcal{D}_i$ with different granularity:

$$\mathcal{R}_{j,\ell} = \left\{ v \;\middle|\; \frac{j}{2^\ell} < \Pr[v_i \leq v] \leq \frac{j+1}{2^\ell} \right\}$$

- The d-th test of box i reveals the region of granularity 2^{-d} that contains v'_i:
  Test 1 of box i reveals $j_1 \in \{0, 1\}$ s.t. $v'_i \in \mathcal{R}_{j_1,1}$
  Test 2 of box i reveals $j_2 \in \{2j_1, 2j_1 + 1\}$ s.t. $v'_i \in \mathcal{R}_{j_2,2}$
  Test 3 of box i reveals $j_3 \in \{2j_2, 2j_2 + 1\}$ s.t. $v'_i \in \mathcal{R}_{j_3,3}$
  etc.

- Testing is like a "binary search" over the support regions
- Discrete $\mathcal{D}_i$: To apply the definition of regions, the tests split realizations with the same value into auxiliary ones – exactly the same way as we did for quantiles in the IID-PROPHET problem (c.f. Section 3.1.2)

[Pic: Distribution, Region, another test splits region in the middle, prob. 1/2 to be in the higher subregion]

Example 4. $n = 2$ boxes and $k = 3$ tests, both $\mathcal{D}_1 = \mathcal{D}_2 = \text{Uniform}\{1, 4, 5, 10\}$.

- Apply first test to box 1. Note $\mathcal{R}_{0,1} = \{1, 4\}$ and $\mathcal{R}_{1,1} = \{5, 10\}$. Based on outcome we know $v_1 \sim \text{Uniform}\{1, 4\}$ or $v_1 \sim \text{Uniform}\{5, 10\}$.
- Apply second test to box 2. If the two tests return different regions, we can already identify the box with the best prize.
- Otherwise, either $v_1, v_2 \in \mathcal{R}_{0,1}$ or $v_1, v_2 \in \mathcal{R}_{1,1}$.
- $v_1, v_2 \in \mathcal{R}_{0,1}$: Apply the third test to box 1. If $v_1 \in \mathcal{R}_{0,2} = \{1\}$, then box 2 is optimal. Otherwise $v_1 \in \mathcal{R}_{1,2} = \{4\}$ and box 1 is optimal.
- $v_1, v_2 \in \mathcal{R}_{1,1}$: Apply the third test to box 1. If $v_1 \in \mathcal{R}_{2,2} = \{5\}$, then box 2 is optimal. Otherwise $v_1 \in \mathcal{R}_{3,2} = \{10\}$ and box 1 is optimal.

We always identify a box with an optimal prize. Obviously, this testing policy is optimal. ■

Finding an optimal testing policy in k-TESTMAX can be formulated as an MDP. The exact formulation is left as an exercise. We only observe that the number of possibilities to apply k tests on n boxes is n^k and, thus, prohibitively large.

Rather than finding an optimal testing policy, we are interested in bounding the performance loss of **optimal testing vs. optimal probing**. Consider an instance with n boxes, their distributions, and any value $k \geq 1$. How much expected value do we lose because we cannot probe (and see) but only test the value of a box? It turns out the difference is only a constant factor, so probing is not much more powerful than testing.

Theorem 21. *For any instance with n boxes and any given value k, the expected reward of the optimal policies in k-TESTMAX and k-PROBEMAX differ by at most a constant factor.*

$k > n$ makes no sense for probing. It only gives more power in testing. Hence, w.l.o.g. we restrict attention to $k \leq n$. Also, we prove the theorem only in the **IID-case**, with prizes in all boxes being identically, independently distributed with $\mathcal{D}_i = \mathcal{D}$.

Let OPT_p be the expected reward for the optimal policy of k-PROBEMAX:

- All boxes are IID, no adaptivity, simply open any set of k boxes.
- W.l.o.g. we pick the first k boxes, thus $OPT_p = \mathbb{E}[\max_{i=1}^{k} v_i]$

Example 5. Consider a *gold-nugget distribution*:

- $\Pr[v_i = k] = 1/k$ and $\Pr[v_i = 0] = 1 - 1/k$, hence $\mathbb{E}[v_i] = 1$.
- Probing k entries yields value of $0 \Leftrightarrow$ all opened prizes 0.
- Happens w. prob. $(1 - 1/k)^k \leq 1/\mathsf{e}$. Hence $OPT_p \geq k \cdot (1 - 1/\mathsf{e}) = \Theta(k)$.

What are intuitive testing strategies? **Test k boxes once**.

- Each tested box i in region $\mathcal{R}_{0,1}$ or $\mathcal{R}_{1,1}$.

Algorithm 10: Consecutive testing towards the $(1 - 1/k)$-quantile

```
k' ← 2^⌈log₂ k⌉, i ← 0                          // k' smallest power of 2 with k' ≥ k
while i < n and less than k tests performed do
    i ← i + 1, j ← 0
    repeat
        j ← j + 1
        Apply next test to box i
    until (j = log₂ k') or (v_i ∉ R_{2^j − 1, j}) or (k tests performed in total)
    if j = log₂ k' and v_i ∈ R_{k'−1, log₂ k'} then return box i
return arbitrary tested box
```

- If $v_i \in \mathcal{R}_{0,1}$, then $v_i = 0$ with probability 1. Otherwise, $\mathbb{E}[v_i \mid v_i \in \mathcal{R}_{1,1}] = 2$
- Hence, we can at best ensure an expected value of 2.

[Pic: Gold nugget, regions, conditional expectation]

Test the box with highest conditional expectation.[1]

- First test: box 1. If $v_1 \in \mathcal{R}_{0,1}$, then $v_1 = 0 < 1 \rightarrow$ never test box 1 again, test box 2.
- Suppose otherwise, then $\mathbb{E}[v_1 \mid v_1 \in \mathcal{R}_{1,1}] = 2 > 1 \rightarrow$ continue testing box 1.
- Second test: box 1. $v_1 \in \mathcal{R}_{2,2}$ or $v_i \in \mathcal{R}_{3,2}$.
- If $v_1 \in \mathcal{R}_{2,2}$, then $v_1 = 0 \rightarrow$ never test box 1 again, test box 2.
- Suppose otherwise, then $\mathbb{E}[v_1 \mid v_1 \in \mathcal{R}_{3,2}] = 4 > 1 \rightarrow$ continue testing box 1.
- etc.

Overall, we test box 1 until we know $v_1 = 0$ or $v_1 = k$. Suppose $k = 2^j$ for some $j > 1$, then this requires at most j tests. If $v_1 = k$, box 1 is optimal. Otherwise, start testing box 2, until we know $v_2 = 0$ or $v_2 = k$, and so on.

One can verify that this is indeed the **optimal testing policy** for the gold-nugget distribution when $k = 2^j < n$. How much expected value does it obtain? ■

In Algorithm 10 we generalize the consecutive testing policy from the example to the entire IID case. It is very closely related to Algorithm 5 for the IID-Prophet problem.

Some observations:

- The algorithm picks k' as a power of 2 such that $k \leq k' < 2k$
- It tests boxes consecutively. Each box is tested at most $\log_2 k'$ times in a row.
- Every time a box i is tested, it should refine to the better region – first to region $\mathcal{R}_{1,1}$, then to $\mathcal{R}_{3,2}$, then to $\mathcal{R}_{7,3}$, etc.
- Once box is not in the better region, repeat-loop breaks $\rightarrow$ start testing next box
- If box completes $\log_2 k'$ tests with better region, it is chosen.
- If we don't find a box that refines to better region for $\log_2 k'$ many tests or if we run out of tests before finding one, we choose any tested box

[1] Break ties according to smallest box number.

Theorem 22. *Algorithm 10 achieves an expected reward of*

$$\left(1 - \frac{1}{\sqrt[4]{\mathsf{e}}} - o(1)\right) \cdot OPT_p \ ,$$

where the asymptotics is in k (and n, since $k \leq n$).

Proof. A box that gets tested $\log_2 k'$ times and turns out to be in the best region $\mathcal{R}_{k'-1,\log_2 k'}$ is called a **great box**. We choose the first great box that we find.

- A great box has a value in the top $1/k'$-fraction of the support of $\mathcal{D}$
- Put differently, the value of a great box is above the $(1 - 1/k')$-quantile[2] of $\mathcal{D}$.
- Recall one of the main arguments from the proof of Theorem 11:

$$\begin{aligned}\mathbb{E}[v_i \mid v_i \text{ above } (1-1/k')\text{-quantile}] &\geq \mathbb{E}[v_j \mid v_j \text{ optimal among } k' \text{ IID realizations}] \\ &\geq \mathbb{E}[v_j \mid v_j \text{ optimal among } k \text{ IID realizations}] \\ &= \mathbb{E}[\max_{j=1}^{k} v_j] = OPT_p\end{aligned}$$

$\rightarrow$ Every great box has an expected value of at least OPT_p.

The approximation factor w.r.t. OPT_p is given by the probability to find a great box. The next lemma shows that this probability is

$$\alpha = 1 - \frac{1}{\sqrt[4]{\mathsf{e}}} - o(1) \ ,$$

where the asymptotics are in k (and n, since we assume $k \leq n$). This proves the theorem. □

Lemma 12. *The probability that Algorithm 10 runs out of tests before finding a great box is at most $1/\sqrt[4]{\mathsf{e}} + o(1)$.*

Proof. The sequence of tests can be seen as a sequence of Benoulli trials:

- Every test further subdivides the regions by a factor of 2.
- Suppose that ℓ tests were applied and $v_i \in \mathcal{R}_{j,\ell}$
- Now apply the $(\ell+1)$-th test. Result is one of two events: $v_i \in \mathcal{R}_{2j,\ell+1}$ or $v_i \in \mathcal{R}_{2j+1,\ell+1}$
- If $v_i \in \mathcal{R}_{2j+1,\ell+1}$ we say the test is a **success**, otherwise a **failure**
- Regions have same probability mass $\rightarrow$ conditional probability of each event is $1/2$.
- Conditional probability does not depend on condition $\rightarrow$ success and failure events constitute a sequence of **independent Bernoulli trials**!

The algorithm can apply $k \leq n$ tests. Upon each failure, it starts testing a new box. A great box is found $\Leftrightarrow$ there is a consecutive sequence of at least $r = \log_2(k')$ successes in a sequence of k independent Benoulli trials.

[Pic: Algorithm 10 as Markov chain, Bernoulli trials, run length]

To avoid trivialities, we assume $\log_2 k' > 1$ and, thus $k > 2$. Each trial has success probability $1/2$. Classic results in probability theory show that:

[2]Note that for discrete distributions, we assume tests work using auxiliary realizations to formally ensure existence of such a quantile, similar to our adjustments in Section 3.1.2.

1. The probability of **no success run of length** $r = \log_2 k'$ (i.e., no great box found) is

$$q = A_1 + A_2 + \ldots A_r \ ,$$

where

$$A_1 = \frac{2-x}{r+1-rx} \cdot \frac{1}{x^{k+1}} \quad \text{and} \quad |A_i| \leq \frac{4}{2^{k+1}3r} \ ,$$

for all $i = 2, \ldots, r = \log_2 k'$.
2. Here x is the root with smallest absolute value of $f(y) = 1 - y + \left(\frac{y}{2}\right)^{r+1}$.
3. $f(y)$ has a single positive root different from 2. This is the root x with smallest absolute value.

In Lemma 13 below we show that $1+\frac{1}{2k'} \leq x \leq 1+\frac{1}{k'}$. This allows to conclude, for all $k > 1$,

$$\begin{aligned}
q &= A_1 + (r-1)\frac{4}{2^{k+1}3r} \\
&= \frac{2-x}{r+1-rx} \cdot \frac{1}{x^{k+1}} + \frac{r-1}{2^{k-1}3r} \\
&\leq \frac{1-\frac{1}{2k'}}{1-\frac{\log_2 k'}{k'}} \cdot \frac{1}{(1+\frac{1}{2k'})^{k+1}} + o(1) && \text{plugging in bounds for } x \text{ and } r \\
&\leq (1+o(1)) \cdot \frac{1}{(1+\frac{1}{2k'})^{(2k'+1)\frac{k+1}{2k'+1}}} + o(1) && \text{plugging in } r = \log_2 k' \\
&\leq (1+o(1)) \cdot \frac{1}{\mathsf{e}^{\frac{k+1}{2k'+1}}} + o(1) && \text{since } (1+1/z)^{z+1} > \mathsf{e} \text{ for all } z > 0 \\
&\leq (1+o(1)) \cdot \frac{1}{\sqrt[4]{\mathsf{e}}} + o(1) && \text{since } (k+1)/(2k'+1) > k/(2k') \geq 1/4
\end{aligned}$$

□

Lemma 13. *Let $r \in \mathbb{N}_{\geq 2}$, and x_0 be the unique positive root of $f(y) = 1 - y + \left(\frac{y}{2}\right)^{r+1}$ that is different from 2. Then, $1 + \frac{1}{2^{r+1}} < x_0 < 1 + \frac{1}{2^r}$.*

Proof. $f(y)$ has a unique positive root that is different from 2. We show that $f(1+\frac{1}{2^{r+1}}) > 0$, and $f(1+\frac{1}{2^r}) < 0$ for all $r \in \mathbb{N}_{\geq 2}$. Since f is continuous, there must be a root of f in between.

First, for the smaller value

$$f\left(1+\frac{1}{2^{r+1}}\right) = 1 - \left(1+\frac{1}{2^{r+1}}\right) + \frac{1}{2^{r+1}}\left(1+\frac{1}{2^{r+1}}\right)^{r+1} > -\frac{1}{2^{r+1}} + \frac{1}{2^{r+1}} \cdot 1 = 0 \ .$$

Second, we assume $r \geq 3$ and observe for the larger value

$$\begin{aligned}
f\left(1+\frac{1}{2^r}\right) &= 1 - \left(1+\frac{1}{2^r}\right) + \frac{1}{2^{r+1}}\left(1+\frac{1}{2^r}\right)^{r+1} \\
&= -\frac{1}{2^r} + \frac{1}{2^{r+1}}\left(1+\frac{1}{2^r}\right)^{r}\left(1+\frac{1}{2^r}\right)
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{2^{r+1}}\left(-2+\left(1+\frac{1}{2^r}\right)^r\left(1+\frac{1}{2^r}\right)\right)\\
&< \frac{1}{2^{r+1}}\left(-2+\mathsf{e}^{\frac{r}{2^r}}\left(1+\frac{1}{2^r}\right)\right)\\
&< \frac{1}{2^{r+1}}\left(-2+\mathsf{e}^{\frac{3}{8}}\left(1+\frac{1}{2^3}\right)\right) \qquad \text{since } r/(2^r) < 3/8 \text{ when } r \geq 3\\
&< \frac{1}{2^{r+1}}\left(-2+2\right) = 0\ .
\end{aligned}
$$

For $r = 2$, the same holds, since $f\left(1+\frac{1}{4}\right) = -3/512 < 0$. □

4.3 Probing with Cost

We consider the Pandora Box problem, a monetary variant of probing:

- n boxes, in each box a stochastic, non-negative prize $v_i \sim \mathcal{D}_i$
- Each box has an **opening cost** $c_i \geq 0$.
- Cost c_i and distribution $\mathcal{D}_i$ are known in advance
- We can open any subset of boxes (adaptively).
- Given a set of open boxes, we can pick a single prize of any opened box.
- **Goal**: Maximize chosen prize minus sum of opening costs:

$$\max_{i \text{ open}} v_i - \sum_{i \text{ open}} c_i$$

This variant captures many economic investment problems, where we want to maximize income minus investment. Opening cost can be interpreted as an "exploration cost" e.g., cost of market exploration (e.g., customer demand for a new tech gadget, or interest in advertising services, etc.) or assessment of production cost/benefit (e.g., when drilling for mineral resources in an area, etc.). Such assessments are usually conducted before engaging into the actual activity (e.g., producing the gadget, drilling in the area, etc.)

The name Pandora Box comes from a work by Weizman on the problem. The reference to Greek mythology is maybe debatable. In any case, one might indeed regret opening a box – it costs you money, but eventually you might learn that the box only offers you a small prize inside. If you find a larger prize later on, opening the box was worthless.

Pandora Box can be formulated as an MDP with a huge state space (details as exercise). Nevertheless, in this section we will design an optimal policy, which can be computed quickly and represented compactly.

Example 6. There are $n = 2$ boxes. Box 1 has prize $v_1 = 4$ with probability $1/2$ and $v_1 = 0$ otherwise. Box 2 has prize $v_2 = 2$ with probability 1. Opening costs are $c_1 = c_2 = 1$. An optimal policy:

- Open box 1. If $v_1 = 4$, stop opening boxes, choose box 1. Reward: $4 - 1 = 3$
- Otherwise, open box 2, choose box 2. Reward: $2 - 2 = 0$
- Overall expected reward: $3 \cdot (1/2) + 0 \cdot (1/2) = 1.5$

Algorithm 11: Pandora Box using Fair Caps

```
Compute the fair cap ρ_i^f for each box
Sort boxes in non-increasing order of ρ_i^f
i* ← 1, i ← 0
repeat
    i ← i + 1
    open box i
    if v_i > v_{i*} then i* ← i
until (i = n) or (v_{i*} ≥ ρ_{i+1}^f)
return box i*
```

Why is this optimal? Suppose we instead open box 2 first. If we stop, reward is 1. If we continue to open box 1, expected reward is also 1. ■

We first consider a single box – when should we open it? Clearly, if $c_i \geq \mathbb{E}[v_i]$ the prize does not justify the investment. This holds also if there are several boxes. Hence, w.l.o.g. $\mathbb{E}[v_i] \geq c_i$ for all boxes (if not, ignore the box).

There is an investor that offers a deal:

- If we open a box, investor pays opening cost c_i, but keeps some part of the prize – everything above a cap ρ_i (defined below)
- Investor gets bonus above cap: $b_i = \max\{0, v_i - \rho_i\}$
- We get value up to the cap: $\kappa_i = \min\{v_i, \rho_i\} = v_i - b_i$.
- What is a good deal for the investor? Depends on the cap ρ_i
- A **fair cap** ρ_i^f yields $\mathbb{E}[b_i] = c_i$. Then expected investor utility is $\mathbb{E}[b_i] - c_i = 0$.
- Fair cap always exists:
  $\rho_i = 0 \Rightarrow \mathbb{E}[b_i] = \mathbb{E}[v_i] \geq c_i$, and $\rho_i = v_i \Rightarrow \mathbb{E}[b_i] = 0 \leq c_i$
  $\mathbb{E}[b_i]$ continuous in ρ_i, so fair cap exists.

Example 6 (continued)**.** Consider box 1.

- If $\rho_1 \geq 4$, then $\kappa_1 = v_1$ and $b_1 = 0$.
- Otherwise, $\kappa_1 = \rho_1$ and $b_1 = 4 - \rho_1$ when $v_1 = 4$, and $\kappa_1 = b_1 = 0$ when $v_1 = 0$.
  So then $\mathbb{E}[\kappa_1] = \rho_1/2$ and $\mathbb{E}[b_1] = (4 - \rho_1)/2 = 2 - \rho_1/2$.
- Fair cap $\rho_1^f = 2$, since then $\mathbb{E}[b_1] = 2 - 1 = c_1$.

■

Algorithm 11 opens boxes in non-increasing order of fair cap ρ_i^f. It stops when (all boxes are open, or) the best current prize v_{i^*} is at least the best fair cap of any closed box.

ρ_i^f is an estimate of the prize we get from box i after subtraction of the cost c_i. We start with the most promising box and continue until the expected gain from opening a (most promising) remaining box is 0.

Theorem 23. *Algorithm 11 computes an optimal policy for the* Pandora Box *problem.*

First, consider an arbitrary policy π. We use two sets of indicator variables for π:

- $I_i = 1$ if π opens box i (i.e., (I)nspects the box)
- $A_i = 1$ if π chooses the prize in box i (i.e., (A)ccepts the box)

Let us formulate the expected reward of π. It turns out that the expected reward is the capped value of the accepted box minus the bonus of every inspected (non-accepted) box.

Lemma 14. *The expected reward of any policy π is*

$$\sum_{i\in[n]} \mathbb{E}[A_i\kappa_i - (I_i - A_i)b_i]$$

Proof. Note that the reward is

$$\begin{aligned}
&\mathbb{E}\left[\sum_{i\in[n]}(A_iv_i - I_ic_i)\right] && \text{by definition}\\
&= \sum_{i\in[n]}(\mathbb{E}[A_iv_i] - \mathbb{E}[I_i]c_i) && \text{linearity of expectation}\\
&= \sum_{i\in[n]}(\mathbb{E}[A_i\kappa_i] + \mathbb{E}[A_ib_i] - \mathbb{E}[I_i]c_i) && \text{since } v_i = \kappa_i + b_i, \text{ and linearity of expectation}\\
&= \sum_{i\in[n]}(\mathbb{E}[A_i\kappa_i] + \mathbb{E}[A_ib_i] - \mathbb{E}[I_i]\mathbb{E}[b_i]) && \rho_i^f \text{ is fair cap} \Rightarrow c_i = \mathbb{E}[b_i]\\
&= \sum_{i\in[n]}(\mathbb{E}[A_i\kappa_i] + \mathbb{E}[A_ib_i] - \mathbb{E}[I_ib_i]) && \text{since } I_i \text{ and } b_i \text{ are independent}
\end{aligned}$$

To see the last equation, b_i depends only on the fixed value ρ_i^f and the random value v_i. However, I_i cannot depend on v_i – we only see the realized value *after* we inspect the box. Hence, I_i and b_i are independent, so $\mathbb{E}[I_i]\mathbb{E}[b_i] = \mathbb{E}[I_ib_i]$. So finally, by linearity of expectation, the expected reward is

$$\sum_{i\in[n]}(\mathbb{E}[A_i\kappa_i] + \mathbb{E}[A_ib_i] - \mathbb{E}[I_ib_i]) = \sum_{i\in[n]}\mathbb{E}[A_i\kappa_i - (I_i - A_i)b_i]$$

□

Now consider Algorithm 11. We prove optimality in two steps in the subsequent lemmas.

Lemma 15. *Algorithm 11 always selects the box of highest capped value. That is, with probability 1*

$$\sum_i A_i\kappa_i = \max_i \kappa_i.$$

Proof. The key insight is that over the course of the loop, the best observed prize v_{i^*} is only increasing, while the maximum fair cap of a closed box ρ_{i+1}^f is only decreasing.

i_{last} is index of last opened box, $i^* \le i_{last}$ box of accpeted prize. We want to show $\kappa_i \le \kappa_{i^*}$ for all i. Consider two cases: The accepted prize exceeds its cap or not.

Case $v_{i^*} \le \rho^f_{i^*}$: Then $\kappa_{i^*} = v_{i^*}$. For $i \le i_{last}$ we have

$$\begin{aligned} \kappa_i &\le v_i && \text{by definition} \\ &\le v_{i^*} && \text{since } v_{i^*} \text{ highest prize up to } i_{last} \\ &= \kappa_{i^*} && \text{since } v_{i^*} \le \rho^f_{i^*} \end{aligned}$$

For $i > i_{last}$ we have

$$\begin{aligned} \kappa_i &\le \rho^f_i && \text{by definition} \\ &\le \rho^f_{i_{last}+1} && \text{by monotonicity} \\ &\le v_{i^*} && \text{since we stopped opening boxes} \\ &= \kappa_{i^*} && \text{since } v_{i^*} \le \rho^f_{i^*} \end{aligned}$$

Case $v_{i^*} > \rho^f_{i^*}$: Then $\kappa_{i^*} = \rho^f_{i^*}$. Observe that $i^* = i_{last}$ because $\rho^f_{i^*+1} \le \rho^f_{i^*}$, so we do not open box $i^* + 1$. For $i < i_{last}$, we have

$$\begin{aligned} \kappa_i &\le v_i && \text{by definition} \\ &\le \rho^f_{i_{last}} && \text{since we did not stop opening boxes} \\ &= \kappa_{i^*} && \text{observed above} \end{aligned}$$

For $i > i_{last}$, we have

$$\begin{aligned} \kappa_i &\le \rho^f_i && \text{by definition} \\ &\le \rho^f_{i_{last}} && \text{by monotonicity} \\ &= \kappa_{i^*} && \text{observed above} \end{aligned}$$

□

Lemma 16. *Algorithm 11 fulfills $(I_i - A_i)b_i = 0$ for all $i \in [n]$ with probability 1.*

Proof. If $b_i = 0$, or if $I_i = 0$ (and hence $A_i = 0$), then statement is trivial.
Suppose $I_i = 1$ and $b_i > 0$.

- The policy opens box i with $b_i > 0$. Then $v_i \ge \rho^f_i$, so it is the last box to be opened.
- Box i opened $\Rightarrow$ maxmimum prize in boxes $1, \dots, i-1$ at most ρ^f_i.
- Hence, v_i is highest prize in boxes $1, \dots, i$, and so $A_i = 1$.

□

Proof of Theorem 23. Consider any policy π'. We denote its indicator variables by A'_i and I'_i, for $i \in [n]$. By Lemma 14, the expected reward is

$$\sum_{i\in[n]} \mathbb{E}[A'_i\kappa_i - (I'_i - A'_i)b_i] \le \sum_{i\in[n]} \mathbb{E}[A'_i\kappa_i] \le \mathbb{E}[\max_i \kappa_i]$$

For the policy computed by Algorithm 11, we apply Lemmas 15 and 16 pointwise to the two terms inside the expectation to see that the reward is

$$\sum_{i\in[n]} \mathbb{E}[A_i\kappa_i - (I_i - A_i)b_i] = \mathbb{E}[\max_i \kappa_i]$$

□

Chapter 5

Recommendation

5.1 Bayesian Persuasion

In Internet markets, information is nowadays a critical resource. Large retailers (e.g., Amazon), booking platforms (Booking.com, hotels.de) or review websites (TripAdvisor) issue recommendations for products, services, hotels, restaurants, locations, etc. However, while a customer often strives to make a decision (e.g., book a hotel, buy a product) that gives the best performance/experience, the platform can have very different preferences based on, e.g., provisions by product manufacturers or large service providers.

Consider the following stylized recommendation scenario.

- Two agents called *sender* $\mathcal{S}$ and *receiver* $\mathcal{R}$
- n boxes. Box i contains a prize-pair (s_i, r_i), where $s_i \geq 0$ is the prize for $\mathcal{S}$ and $r_i \geq 0$ the prize for $\mathcal{R}$
- Set Θ of possible prize-pair vectors $\theta_j = ((s_{1j}, r_{1j}), \ldots, (s_{nj}, r_{nj}))$, $j \in [m]$
- Distribution $\mathcal{D}$ over Θ. $\mathcal{D}$ is known to both $\mathcal{S}$ and $\mathcal{R}$ apriori.
- Set Σ of possible *recommendations* or *signals* that $\mathcal{S}$ can send to $\mathcal{R}$.

We study a messaging problem of $\mathcal{S}$ called Persuade.

- $\mathcal{S}$ chooses a *signaling scheme* φ. φ maps every vector θ_j of prize-pairs in the boxes to a distribution over Σ
- Nature draws the prizes in the boxes $\theta \sim \mathcal{D}$
- $\mathcal{S}$ sees θ. $\mathcal{R}$ does not see θ, knows only φ and $\mathcal{D}$.
- $\mathcal{S}$ sends the (randomized) signal $\sigma = \varphi(\theta)$. $\mathcal{R}$ sees σ and chooses a box i with highest conditional expectation $\mathbb{E}[r_i \mid \sigma]$.
- $\mathcal{S}$ and $\mathcal{R}$ receive their prize in the chosen box.
- **Goal:** Find φ^* that yields highest expected prize s_i in the box chosen by $\mathcal{R}$.

Example 7. Retailer $\mathcal{S}$ wants to sell a washing machine to customer $\mathcal{R}$. There are two models: a fancy one (box 1) and a solid one (box 2). The fancy one is either great or mediocre, both with prob. 1/2. The manufacturer pays $\mathcal{S}$ a provision if she sells the fancy one. If great and $\mathcal{R}$ picks it, then $s_2 = 10$ and $r_2 = 3$. Otherwise, $s_2 = 10$ and $r_2 = 0$. The solid one has values $s_1 = r_1 = 2$ with prob. 1. Hence, Θ is composed of $\theta_1 = ((10, 3), (2, 2))$ and $\theta_2 = ((10, 0), (2, 2))$, and in $\mathcal{D}$ both have prob. 1/2.

In the beginning, $\mathcal{S}$ decides how to send recommendations. Suppose she commits to reveal the quality of machine 1 truthfully:

- $\mathcal{S}$ sees θ. If $\theta = \theta_1$, fancy machine is great, $\mathcal{S}$ sends $\varphi(\theta_1) = 1$. Otherwise $\varphi(\theta_2) = 2$.
- For $\mathcal{R}$ the recommended one is always the best. He knows this and picks it.

$\Rightarrow$ Expected reward for $\mathcal{S}$ is $10 \cdot 1/2 + 2 \cdot 1/2 = 6$.

Is this optimal for $\mathcal{S}$? She wants to sell the fancy machine even if it is mediocre. Suppose she always recommends it, $\varphi(\theta_1) = \varphi(\theta_2) = 1$.

- $\mathcal{S}$ sees θ and sends signal 1. Since signal is always 1, it contains no information for $\mathcal{R}$.
- $\mathcal{R}$'s choice reduces to compare his unconditional expectations.
- Fancy machine has only expected value $1.5 < 2$, $\mathcal{R}$ always picks solid machine.

$\Rightarrow$ Expected reward for $\mathcal{S}$ is 2.

Another scheme: $\varphi(\theta_1) = 1$, but $\varphi(\theta_2) = 1$ with prob. $1/2$ and $\varphi(\theta_2) = 2$ otherwise.

- $\mathcal{S}$ sees θ and uses φ to generate the recommendation.
- $\mathcal{R}$ sees $\varphi(\theta) = 1 \rightarrow$ happens in θ_1 (always), or in θ_2 w.prob. $1/2$.
  Expected utilities of $\mathcal{R}$:

$$\begin{aligned}\mathbb{E}[r_{1j} \mid \varphi(\theta) = 1] &= \frac{\Pr[\theta = \theta_1] \cdot \Pr[\varphi(\theta_1) = 1] \cdot r_{11} + \Pr[\theta = \theta_2] \cdot \Pr[\varphi(\theta) = 1] \cdot r_{12}}{\Pr[\varphi(\theta) = 1]} \\ &= \frac{\frac{1}{2} \cdot 1 \cdot 3 + \frac{1}{2} \cdot \frac{1}{2} \cdot 0}{\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \frac{1}{2}} \\ &= \frac{\frac{3}{2}}{\frac{1}{2} + \frac{1}{4}} = 2\end{aligned}$$

  and

$$\begin{aligned}\mathbb{E}[r_{2j} \mid \varphi(\theta) = 1] &= \frac{\Pr[\theta = \theta_1] \cdot \Pr[\varphi(\theta_1) = 1] \cdot r_{21} + \Pr[\theta = \theta_2] \cdot \Pr[\varphi(\theta) = 1] \cdot r_{22}}{\Pr[\varphi(\theta) = 1]} \\ &= \frac{\frac{1}{2} \cdot 2 + \frac{1}{4} \cdot 2}{\frac{1}{2} + \frac{1}{4}} = 2\end{aligned}$$

- Break ties in favor of $\mathcal{S}$: Machine 1 is an optimal choice for $\mathcal{R}$.
- $\mathcal{R}$ sees $\varphi(\theta) = 2 \rightarrow$ happens only in θ_2, so 2 is unique optimal choice for $\mathcal{R}$.

$\Rightarrow$ Expected reward for $\mathcal{S}$ is $10 \cdot 1/2 + 10 \cdot 1/4 + 2 \cdot 1/4 = 8$

It can be shown that this is indeed the optimal signaling scheme for $\mathcal{S}$. ■

Upon recommendation "1", both machines have the same conditional expectation for $\mathcal{R}$. $\mathcal{S}$ could add tiny ε in φ to induce a strict preference for machine 1 for $\mathcal{R}$. We avoid messing around with ε's – instead always assume $\mathcal{R}$ **breaks ties in favor of** $\mathcal{S}$.

While the space of signals Σ is arbitrary, we can w.l.o.g. restrict attention to signaling schemes, in which $\mathcal{S}$ just recommends a single box, such that $\mathcal{R}$ wants to take that box.

A signaling scheme is called **direct** if every signal just recommends taking a single box, i.e., $\Sigma = [n]$. A direct scheme is called **persuasive** if the recommended box always gives the best expected reward for $\mathcal{R}$.

Proposition 3. *For every signaling scheme φ there is a direct and persuasive scheme φ' such that $\mathcal{S}$ obtains the same expected value in φ and φ'.*

Proof. Exercise. □

If the set Θ of m prize-pair vectors is represented explicitly, finding an optimal signaling scheme φ^* can be done in polynomial time.

Theorem 24. *An optimal signaling scheme φ^* can be obtained by solving a linear program with nm variables and $O(n(n+m))$ constraints.*

Proof. We concentrate on finding a direct and persuasive scheme φ.

- φ yields a distribution over $\Sigma = [n]$ for every possible vector of prizes θ_j, $j \in [m]$.
- If signal $\sigma \in [n]$ is sent, then $\mathcal{R}$ must finds it in his interest to take box σ.
- Notation: $q_j = \Pr[\theta = \theta_j]$ for the prob. that θ_j is drawn and $x_{ij} = \Pr[\varphi(\theta_j) = i]$ for the prob. that φ issues signal i in state θ_j.
- The expected reward of $\mathcal{S}$ is
$$\sum_{j\in[m]} q_j \sum_{i\in[n]} s_{ij} \cdot x_{ij}$$
- φ must be persuasive, so $\mathcal{R}$ must find it in his interest to follow the recommendation. Hence
$$\mathbb{E}[r_{ij} \mid \varphi(\theta) = i] \geq \mathbb{E}[r_{i'j} \mid \varphi(\theta) = i]$$
where the conditional expectations are
$$\frac{\sum_{j\in[m]} q_j \cdot x_{ij} \cdot r_{ij}}{\Pr[\varphi(\theta) = i]} \geq \frac{\sum_{j\in[m]} q_j \cdot x_{ij} \cdot r_{i'j}}{\Pr[\varphi(\theta) = i]}$$
- If $\Pr[\varphi(\theta) = i] > 0$, this is equivalent to
$$\sum_{j\in[m]} q_j r_{ij} \cdot x_{ij} \geq \sum_{j\in[m]} q_j r_{i'j} \cdot x_{ij} \tag{5.1}$$
Otherwise, if $\Pr[\varphi(\theta) = i] = 0$, then signal i is never issued. Then $q_j x_{ij} = 0$ for all $j \in [m]$ and hence $\sum_{j\in[m]} q_j r_{i'j} x_{ij} = 0$ for all boxes $i' \in [n]$. As such, (5.1) still holds.

Overall, finding a scheme φ is equivalent to solving the LP

$$\begin{aligned}
\text{Maximize} \quad & \sum_{j\in[m]} \sum_{i\in[n]} q_j s_{ij} \cdot x_{ij} \\
\text{subject to} \quad & \sum_{j\in[m]} q_j r_{ij} \cdot x_{ij} \;\geq\; \sum_{j\in[m]} q_j r_{i'j} \cdot x_{ij} && \forall i, i' \in [n] \\
& \sum_{i\in[n]} x_{ij} \;=\; 1 && \forall j \in [m] \\
& x_{ij} \;\geq\; 0 && \forall i \in [n], j \in [m].
\end{aligned} \tag{5.2}$$

□

5.1.1 IID Boxes

In many interesting domains, the number of possible prize vectors for the boxes is exponential. We first consider Persuade with IID-Boxes. Here, we have a distribution *for the prize-pairs in each box*, and boxes are IID distributed.

- A single distribution $\mathcal{D}$ over a set $\Theta = \{(s_j, r_j) \mid j \in [m]\}$ with m possible prize pairs.
- Θ and $\mathcal{D}$ describe the **possible prize-pairs of one box** and their probabilities.
- The prize-pair in each box is drawn **independently** from $\mathcal{D}$.
- Notation: $q_j = \Pr[\theta = \theta_j]$, the probability of prize-pair θ_j in $\mathcal{D}$.

Note that even if distribution $\mathcal{D}$ has only $|\Theta| = 2$ possible outcomes for the pair in each box i, the number of possible prize vectors for the n boxes is as large as 2^n.

We cannot explicitly write down LP (5.2) or an optimal scheme φ^* in polynomial time. Still, there is a (rather intricate) way to implicitly compute and represent an optimal scheme φ^*. We here consider a much more simple approach that manages to obtain a good approximately-optimal solution. Our algorithm will have similarities with algorithms for Prophet and k-TestMax problems in the IID case. Overall, the analysis approach again uses LPs to overestimate the optimum, as we did in Section 4.1.

We analyze the structure of φ^* and design a simpler and smaler LP using only a subset of constraints that φ^* must satsify. The optimal LP value is an upper bound for the expected reward of φ^*. Algorithm 12 given below then turns the LP-optimum into a signaling scheme.

Note that there is an optimal scheme φ^* that is **symmetric**, i.e., it satisfies

$$\Pr[\varphi^*(\theta_1, \dots, \theta_n) = i] = \Pr[\varphi^*(\theta_{\pi(1)}, \dots, \theta_{\pi(n)}) = \pi(i)]$$

for every $i \in [n]$ and every permutation π. Intuitively, if the same set of realizations occur in a different permutation in the boxes, we also permute the signal distributions in φ^* accordingly. This is natural, since all permutations of the prize pairs have the same probability to arise in the boxes.

Lemma 17. *There is an optimal signaling scheme φ^* for* Persuade with IID-Boxes *that is symmetric.*

Proof. Consider any optimal scheme φ^*. Now suppose there is a hurricane: Nature permutes all boxes in the beginning uniformly at random before $\mathcal{S}$ can look into them. Then, since the boxes are IID, φ^* applied to the permuted box set remains direct and persuasive and obtains the same optimal expected reward for $\mathcal{S}$.

We simply make this permutation step part of our scheme! Let's extend φ^* to φ' by an initial, uniform random permutation of the boxes. Then, clearly, the scheme is direct, persuasive, obtains an optimal expected reward for $\mathcal{S}$, and also satisfies

$$\begin{aligned}\Pr[\varphi'(\theta_1, \dots, \theta_n) = i] &= \frac{1}{n!} \sum_{\text{Permutation } \pi'} \Pr[\varphi^*(\theta_{\pi'(1)}, \dots, \theta_{\pi'(n)}) = \pi'(i)] \\ &= \Pr[\varphi'(\theta_{\pi(1)}, \dots, \theta_{\pi(n)}) = \pi(i)]\end{aligned}$$

□

Symmetry of φ^* has drastic consequences. From the perspective of $\mathcal{R}$, φ^* yields only two conditional distributions: One for the prize pairs in a *recommended box* i (no matter which one), and one for the prize pairs in a *non-recommended box* i' (no matter which box i was recommended and which box $i' \neq i$ is chosen).

Lemma 18. *Any symmetric optimal scheme φ yields a single conditional distribution for every recommended box $i \in [n]$. It yields a single conditional distribution for every non-recommended box $i' \in [n]$, no matter which box $i \neq i'$ is recommended.*

Proof. Consider a symmetric φ and assume that a specific prize pair is in box $i \in [n]$.

- For each $\theta_j \in \Theta$ and all $i \in [n]$ we denote by

$$x_{ij} = \Pr[\theta_j \text{ in box } i \wedge \varphi \text{ signals } i].$$

- By permuting the prizes in all boxes, we see that for each pair of boxes $i, i' \in [n]$

$$x_{ij} = x_{i'j} = x_j.$$

  This shows that, no matter what box i is recommended,

$$\Pr[\theta_j \text{ in box } i \mid \varphi \text{ signals } i] = \frac{\Pr[\theta_j \text{ in box } i \wedge \varphi \text{ signals } i]}{\Pr[\varphi \text{ signals } i]} = n \cdot x_j$$

  since $\Pr[\varphi \text{ signals } i] = 1/n$ by symmetry $\Rightarrow$ First statement of the lemma follows.
- For any $i \neq i'$, we denote by

$$y_{iji'} = \Pr[\theta_j \text{ in box } i \wedge \varphi \text{ signals } i'].$$

- Due to above, the probability that q_j is in box i and the signal does not yield box i is $q_j - x_{ij} = q_j - x_j$. By symmetry, for all other boxes $i', i'' \neq i$

$$y_{iji'} = y_{iji''} = \frac{q_j - x_j}{n-1} \ .$$

  Symmetry also implies $y_{iji'} = y_{i''ji'}$ for all $i, i'' \neq i'$. Hence, the conditional probability

$$\Pr[\theta_j \text{ in box } i' \mid \varphi \text{ signals } i] = \frac{n}{n-1}(q_j - x_j)$$

  for all $i \neq i'$ and the second statement follows.

□

We build an LP using notation from the proof of Lemma 18 for an optimal symmetric φ^*.

- The expected reward of φ^* is the expected value of $\mathcal{S}$ for a recommended box:

$$\sum_{i \in [n]} \sum_{j \in [m]} x_{ij} s_j = \sum_{j \in [m]} n s_j \cdot x_j$$

- Clearly, $0 \leq x_j \leq q_j$ by definition. Symmetry implies for every box i

$$\Pr[\varphi^* \text{ signals } i] = \sum_{j \in [m]} \Pr[\theta_j \text{ in box } i \wedge \varphi \text{ signals } i] = \sum_{j \in [m]} x_j = \frac{1}{n} \ .$$

Algorithm 12: $(1-1/\mathsf{e})^{-1}$-approximation for PERSUADE WITH IID-BOXES

1 Solve LP 5.3, let x^* be an optimal solution
2 **for** *each box* $i = 1, \dots, n$ **do**
3 Let $\theta_j = (s_j, r_j)$ be realized prize-pair in box i
4 With prob. x^*_j/q_j label box i "yes", otherwise label i "no"
5 **if** $\exists$ *at least one yes-box* **then return** uniform random yes-box
6 **else return** uniform random no-box

- Lemma 18 and persuasiveness yield the following constraint

$$\mathbb{E}[r_i \mid \varphi^* \text{ signals } i] = \sum_{j\in[m]} nx_j \cdot r_j \geq \sum_{j\in[m]} \frac{n}{n-1}(q_j - x_j) \cdot r_j = \mathbb{E}[r_{i'} \mid \varphi^* \text{ signals } i]$$

Multiplying by $n/(n-1)$ implies

$$\sum_{j\in[m]} (n-1)x_j r_j \geq \sum_{j\in[m]} (q_j - x_j) r_j,$$

and adding $\sum_{j\in[m]} x_j r_j$ we obtain

$$\sum_{j\in[m]} nr_j \cdot x_j \geq \sum_{j\in[m]} q_j r_j = r_E,$$

where r_E is the (unconditional) expected reward of any box for $\mathcal{R}$.

We collect the previous observations and formulate the following LP.

$$\begin{aligned} \text{Maximize} \quad & \sum_{j\in[m]} ns_j \cdot x_j \\ \text{subject to} \quad & \sum_{j\in[m]} nr_j \cdot x_j \;\geq\; r_E \\ & \sum_{j\in[m]} x_j \;=\; \frac{1}{n} \\ & x_j \;\in\; [0, q_j] \quad \forall j \in [m] \end{aligned} \tag{5.3}$$

Some observations:

- The second constraint mirrors the idea of a $(1-1/n)$-quantile – the combination of realization and recommendation yields only a $1/n$-fraction of probability mass.
- However, the LP is not simply taking a best $1/n$-fraction of $\mathcal{D}$ that maximizes prizes for $\mathcal{S}$ – it determines a $1/n$-fraction that maximizes the value for $\mathcal{S}$ *and guarantees at least an average value for* $\mathcal{R}$.
- If we set x_j according to φ^*, we obtain a feasible solution for the LP.

Algorithm 12 solves the LP optimally. Let x^* be an LP-optimum. Note that the rounding in line 4 is possible, since $x^*_j \in [0, q_j]$ due to the last constraints of LP (5.3).

Similar as for the LP in Section 4.1, we observe that the value of x^* can be strictly more than the expected reward of the optimal signaling scheme φ^*: There are $n = 2$ boxes, and the distribution over prize-pairs in each box is $\theta_1 = (1, 1)$ and $\theta_2 = (0, 0)$, each with probability $1/2$. In φ^*, we signal any box that has value θ_1. If both boxes have the same value, we choose a box at random. Clearly, this is direct and persuasive. The expected value for $\mathcal{S}$ (and $\mathcal{R}$) is $3/4$. Note that the optimal LP solution sets $x^*_1 = 1/2$ and $x^*_2 = 0$ and obtains a value of $2 \cdot 1/2 \cdot 1 + 2 \cdot 0 \cdot 0 = 1$.

Theorem 25. *Algorithm 12 computes a direct and persuasive signaling scheme that obtains a $(1 - 1/\mathsf{e})^{-1}$-approximation of the optimal expected reward of $\mathcal{S}$ for* Persuade with IID-Boxes.

Proof. We need to (a) bound the value of the scheme computed by Algorithm 12 and (b) show that it is persuasive. Towards (a), the probability that box i is a yes-box is, by LP (5.3)

$$\Pr[i \text{ yes-box}] = \sum_{j \in [m]} q_j \cdot \frac{x^*_j}{q_j} = \sum_{j \in [m]} x^*_j = \frac{1}{n} \ .$$

Note that

$$\Pr[\theta_j \text{ in box } i \mid i \text{ yes-box}] = \frac{\Pr[\theta_j \text{ in box } i \wedge i \text{ yes-box}]}{\Pr[i \text{ yes-box}]} = \frac{q_j \cdot (x^*_j/q_j)}{1/n} = n \cdot x^*_j, \tag{5.4}$$

so every box has the same distribution conditioned on being a yes-box. Note that every box also has the same distribution conditioned on being a no-box:

$$\Pr[\theta_j \text{ in box } i \mid i \text{ no-box}] = \frac{\Pr[\theta_j \text{ in box } i \wedge i \text{ no-box}]}{\Pr[i \text{ no-box}]} = \frac{q_j \cdot (1 - x^*_j/q_j)}{1 - 1/n} = \frac{n}{n-1} \cdot (q_j - x^*_j). \tag{5.5}$$

Becoming a yes- or no-box is an independent event for each box, so with probability $p_{yes} = 1 - (1 - 1/n)^n$ there is at least one yes-box. We pick uniformly at random among the yes- (if available) and no-boxes (otherwise). Hence, the signaling scheme computed by Algorithm 12 is symmetric, and Lemma 18 applies.

Suppose $\mathcal{R}$ takes any recommended box. The expected reward for $\mathcal{S}$ due to yes-boxes is

$$p_{yes} \cdot \sum_{j \in [m]} n \cdot x^*_j \cdot s_j.$$

i.e., a p_{yes}-fraction of the optimal LP value. This proves the approximation ratio, since $p_{yes} \geq 1 - 1/\mathsf{e}$ and the optimal LP value is more than the expected reward of φ^*.

Towards (b) and persuasiveness, the computed scheme has one conditional distribution for every recommended box and one for every non-recommended box. Notation:

- r^+_E = expected value of $\mathcal{R}$ for any recommended box
- r^-_E = expected value of $\mathcal{R}$ for any non-recommended box

Recall $r_E = \sum_j q_j r_j$. With probability p_{yes} the recommended box is a yes-box, otherwise a no-box. Hence, by (5.4) and (5.5)

$$\begin{aligned}
r_E^+ &= p_{yes} \sum_{j\in[m]} n x_j^* r_j + (1-p_{yes}) \sum_{j\in[m]} \frac{n}{n-1} \cdot (q_j - x_j^*) r_j \\
&= \left(p_{yes} - \frac{1-p_{yes}}{n-1} \right) \sum_{j\in[m]} n x_j^* r_j + (1-p_{yes}) \cdot \frac{n}{n-1} \cdot r_E \\
&\geq \left(p_{yes} - \frac{1-p_{yes}}{n-1} \right) \cdot r_E + (1-p_{yes}) \cdot \frac{n}{n-1} \cdot r_E \\
&= \frac{1}{n-1} \left((n-1)p_{yes} - 1 + p_{yes} + n - n p_{yes} \right) r_E \\
&= r_E
\end{aligned}$$

Now consider box 1. By symmetry, it is recommended with probability $1/n$, in which case it has expected value r_E^+, and r_E^- otherwise. Overall, the expected value of box 1 is

$$r_E = \frac{1}{n} \cdot r_E^+ + \left(1 - \frac{1}{n}\right) \cdot r_E^- \geq \frac{1}{n} \cdot r_E + \left(1 - \frac{1}{n}\right) \cdot r_E^-,$$

which implies $\left(1 - \frac{1}{n}\right) r_E \geq \left(1 - \frac{1}{n}\right) r_E^-$, and, thus, $r_E^+ \geq r_E \geq r_E^-$. This shows that it is better for $\mathcal{R}$ to pick any recommended box than to pick any non-recommended box. □

5.1.2 Independent Boxes

We generalize the previous arguments to independent (not necessarily identical) distributions. For the problem Persuade with Independent Boxes we use a very similar notation as above.

- For every box i there is a distribution $\mathcal{D}_i$ over a set $\Theta_i = \{(s_{ij}, r_{ij}) \mid j \in [m_i]\}$ with m_i possible prize pairs
- Θ_i and $\mathcal{D}_i$ describe possible prize-pairs **of box** i and their probabilities
- Prize-pair in each box is drawn **independently** from $\mathcal{D}_i$
- Notation: $q_{ij} = \Pr[\theta_i = \theta_{ij}]$, the probability of prize-pair θ_{ij} in $\mathcal{D}_i$

We make an **additional assumption:** The box i^* with best apriori reward for $\mathcal{R}$ is **deterministic** for $\mathcal{R}$, i.e.,

$$r_{i^*,j} = r_E^{\max} \geq \mathbb{E}[r_i] \qquad \text{for all } j \in [m_{i^*}] \text{ and all } i \in [m]. \tag{5.6}$$

Alternatively, we can assume that $\mathcal{R}$ owns a prize of value $r_E^{\max}$ and must decide if one of the boxes offer him a better deal. We say box i^* is a **satisfactory status quo (SSQ)** box. W.l.o.g. we relabel the boxes to have $i^* = n$.

With SSQ box, the optimal scheme φ^* is not necessarily symmetric, but still we can obtain a constant-factor approximation. We again construct an LP whose optimal solution overestimates the expected reward of an optimal direct and persuasive scheme φ^*. Then φ^* results

in a feasible solution with objective function value equal to the expected reward of φ^*. We solve the LP optimally and round an optimal solution x^* to obtain a direct and persuasive scheme. The rounding deteriorates the expected reward only by a constant factor.

We again use the notation

$$x_{ij} = \Pr[\theta_{ij} \text{ in box } i \wedge \varphi^* \text{ signals } i].$$

To formulate the LP, we use the expected reward as objective function and identify a number of constraints that φ^* must satisfy.

- The expected reward of $\mathcal{S}$ in φ^* is
$$\sum_{i\in[n]} \sum_{j\in[m_i]} s_{ij} \cdot x_{ij}$$
- Clearly, by definition $0 \le x_{ij} \le q_{ij}$. Also, since φ^* is direct, it recommends a box with probability 1, i.e.,
$$\sum_{i\in[n]} \sum_{j\in[m_i]} x_{ij} = 1.$$
- For persuasiveness of φ^*, we require that
$$\begin{aligned}
\mathbb{E}[r_i \mid \varphi^* \text{ signals } i] &= \sum_{j\in[m_i]} r_{ij} \cdot \Pr[\theta_{ij} \text{ in box } i \mid \varphi^* \text{ signals } i] \\
&= \sum_{j\in[m_i]} r_{ij} \cdot \frac{\Pr[\theta_{ij} \text{ in box } i \wedge \varphi^* \text{ signals } i]}{\Pr[\varphi^* \text{ signals } i]} \\
&= \sum_{j\in[m_i]} r_{ij} \cdot \frac{x_{ij}}{\sum_{k\in[m_i]} x_{ik}} \\
&\ge r_E^{\max}
\end{aligned}$$
where $r_E^{\max}$ is the reward of the SSQ box n for $\mathcal{R}$. This is clearly necessary for persuasiveness – $\mathcal{R}$ can always secure a reward of $r_E^{\max}$, since the SSQ box achieves this value deterministically.
- Note that $\sum_{k\in[m_i]} x_{ik} = 0$ implies $\sum_{k\in[m_i]} x_{ij} r_{ij} = 0$ in φ^*. Hence, we know that φ^* actually satisfies
$$\sum_{j\in[m_i]} x_{ij} r_{ij} \ge r_E^{\max} \cdot \sum_{k\in[m_i]} x_{ik}.$$

This leads to the following LP:

$$\begin{aligned}
\text{Maximize} \quad & \sum_{i\in[n]} \sum_{j\in[m_i]} s_{ij} \cdot x_{ij} \\
\text{subject to} \quad & \sum_{j\in[m_i]} r_{ij} \cdot x_{ij} \;\ge\; r_E^{\max} \cdot \sum_{j\in[m_i]} x_{ij} \\
& \sum_{i\in[n]} \sum_{j\in[m_i]} x_{ij} \;=\; 1 \\
& x_{ij} \;\in\; [0, q_{ij}] \qquad\qquad \forall i \in [n], j \in [m_i]
\end{aligned} \tag{5.7}$$

Algorithm 13: 4-approximation for Persuade with Independent Boxes and SSQ box

1 Solve LP 5.7, let x^* be an optimal solution
2 Let the SSQ box be $i^* = n$
3 **for** *each box* $i = 1, \dots, n-1$ **do**
4 Let $\theta_{ij} = (s_{ij}, r_{ij})$ be realized prize-pair in box i
5 With prob. $x^*_{ij}/2q_{ij}$ **return** box i
6 **return** box n

Theorem 26. *Algorithm 13 computes a direct and persuasive signaling scheme that obtains a 4-approximation of the optimal expected reward of $\mathcal{S}$ for* Persuade with Independent Boxes *with SSQ box.*

Proof. Algorithm 13 is an adaptation of Algorithm 12, and we use similar arguments for the analysis. The algorithm considers the non-SSQ boxes **sequentially** in arbitrary order. The rounding step in line 5 works since $x^*_{ij} \in [0, q_{ij}]$ by LP (5.7).

We say box i is a **yes-box** if it gets signaled during the for-loop. We have to show that (1) the resulting scheme obtains a good expected reward for $\mathcal{S}$ and (2) it is pesuasive for $\mathcal{R}$.

For the expected reward, we assume that $\mathcal{R}$ follows the signal. First, consider the event that box i is not reached by the for-loop. Observe that

$$
\begin{aligned}
&\Pr[i \text{ not reached}] \\
&= \Pr[i-1 \text{ not reached} \vee (i-1 \text{ reached} \wedge i-1 \text{ yes-box})] \\
&\le \Pr[i-1 \text{ not reached}] + \Pr[i-1 \text{ reached} \wedge i-1 \text{ yes-box}] && \text{by union bound} \\
&\le \Pr[i-1 \text{ not reached}] + \Pr[i-1 \text{ yes-box}] && \text{by inclusion of events} \\
&\le \Pr[i-1 \text{ not reached}] + \sum_{j \in [m_{i-1}]} q_{i-1,j} \cdot \frac{x^*_{i-1,j}}{2q_{ij}} \\
&= \Pr[i-1 \text{ not reached}] + \frac{1}{2} \sum_{j \in [m_{i-1}]} x^*_{i-1,j} \\
&\le \sum_{i'=0}^{i-1} \frac{1}{2} \sum_{j \in [m_{i'}]} x^*_{i-1,j} && \text{by induction} \\
&\le \frac{1}{2} && \text{by constraint in LP (5.7)}
\end{aligned}
$$

Hence, $\Pr[i \text{ reached}] \ge 1/2$.

It is easy to see[1] that the following events are independent: (a) for-loop reaches box i, (b) box i contains θ_{ij} and (c) box i is a yes-box. As a result, the expected reward of $\mathcal{S}$ from

[1]Note that Algorithm 13 is similar in structure to Algorithm 9 for k-ProbeMax in Section 4.1, so the arguments from the proof of Theorem 20 apply here.

yes-boxes is

$$\sum_{i\in[n]} \mathbb{E}[s_i \mid i \text{ reached} \wedge i \text{ yes-box}] = \sum_{i\in[n]} \Pr[i \text{ reached}] \cdot \sum_{j\in[m_i]} q_{ij} \cdot \frac{x^*_{ij}}{2q_{ij}} \cdot s_{ij} \geq \frac{1}{4} \sum_{i\in[n]} \sum_{j\in[m_i]} x^*_{ij} \cdot s_{ij}.$$

This proves the approximation ratio.

Now consider persuasiveness. Consider the boxes $i < n$. For each such box

- $r_E^{i,+}$ = expected value of $\mathcal{R}$ for box i if it is recommended
- $r_E^{i,-,p}$ = expected value of $\mathcal{R}$ for box i if some box $i' < i$ is recommended
- $r_E^{i,-,s}$ = expected value of $\mathcal{R}$ for box i if some box $i' > i$ is recommended

Suppose box $i < n$ has a positive probability of getting recommended, i.e., for which $\sum_{k\in[m_i]} x^*_{ik} > 0$. Since i is not the SSQ box, it must be a yes-box to get recommended, and

$$\begin{aligned}
r_E^{i,+} &= \mathbb{E}[r_i \mid i \text{ reached} \wedge i \text{ yes-box}] \\
&= \frac{\sum_{j\in[m_i]} \Pr[i \text{ reached, contains } \theta_{ij}, \text{ yes-box}] \cdot r_{ij}}{\sum_{k\in[m_i]} \Pr[i \text{ reached, contains } \theta_{ik}, \text{ yes-box}]} \\
&= \frac{\sum_{j\in[m_i]} \Pr[i \text{ reached}] \cdot q_{ij} \cdot x^*_{ij}/(2q_{ij}) \cdot r_{ij}}{\sum_{k\in[m_i]} \Pr[i \text{ reached}] \cdot q_{ik} \cdot x^*_{ik}/(2q_{ik})} \\
&= \frac{\sum_{j\in[m_i]} x^*_{ij} \cdot r_{ij}}{\sum_{k\in[m_i]} x^*_{ik}} \\
&\geq r_E^{\max} \qquad \text{by constraint in LP (5.7) and } \sum_{k\in[m_i]} x^*_{ik} > 0.
\end{aligned}$$

Now let $p^i_{yes} = \Pr[i \text{ reached} \wedge i \text{ yes-box}]$ the probability that i is recommended. Clearly, if $p^i_{yes} = 1$, then $\mathcal{S}$ always recommends box i, and $\mathcal{R}$ never sees $r_E^{i,-,p}$ or $r_E^{i,-,s}$. Otherwise, when $p^i_{yes} < 1$, then using (5.6)

$$r_E^{i,-,p} = \mathbb{E}[r_i] \leq r_E^{max},$$

since the decision to signal box $i' < i$ in the for-loop is made without even looking at box i.

Now for $i' > i$, we know $r_E^{i,-,s} = \mathbb{E}[r_i \mid i \text{ reached} \wedge i \text{ not yes-box}]$, since the for-loop passed through i. Note that

$$\mathbb{E}[r_i] = \mathbb{E}[r_i \mid i \text{ reached}] = p^i_{yes} \cdot r_E^{i,+} + (1 - p^i_{yes}) r_E^{i,-,s}$$

and, using (5.6),

$$r_E^{i,-,s} = \frac{\mathbb{E}[r_i] - p^i_{yes} \cdot r_E^{i,+}}{1 - p^i_{yes}} \leq \frac{r_E^{\max} - p^i_{yes} \cdot r_E^{i,+}}{1 - p^i_{yes}} \leq \frac{r_E^{\max} - p^i_{yes} \cdot r_E^{\max}}{1 - p^i_{yes}} = r_E^{\max}.$$

Finally, for the SSQ box, we have $r_E^{n,+} = r_E^{n,-,p} = r_E^{\max}$ by (5.6). Hence, choosing any recommended box is always weakly better for $\mathcal{R}$ than choosing any non-recommended box.

□

5.2 Delegation

In the previous section, we assumed that $\mathcal{S}$ has **commitment power**. $\mathcal{S}$ commits in advance on her behavior φ^*, before seeing the actual contents of all boxes. Then, since she is committed, she also sometimes must send signals for boxes that are suboptimal for her. Overall, however, this form of commitment power usually is very beneficial for $\mathcal{S}$.

There is a long discussion in economics, when and which one of the agents actually have commitment power in recommendation scenarios. When the sender is a large retailer like Amazon and the receiver is a single customer, commitment power for the sender seems indeed reasonable. In other applications, one could also imagine that commitment power is with the receiver, most notably in scenarios with **delegation**.

In the Delegation problem, a receiver $\mathcal{R}$ does not want to check all possible decision choices himself. He delegates the search of a good decision to a sender $\mathcal{S}$. Consider, e.g., a company $\mathcal{R}$ hiring a headhunter $\mathcal{S}$ to find a person for a high-profile job. $\mathcal{S}$ inspects all candidates and suggests one to the company $\mathcal{R}$. The company then inspects the candidate herself and decides to accept or reject it.

While $\mathcal{S}$ wants $\mathcal{R}$ to accept a candidate that is good for $\mathcal{S}$, $\mathcal{R}$ wants $\mathcal{S}$ to suggest a candidate that is good for $\mathcal{R}$. Now here we assume $\mathcal{R}$ has commitment power – he commits in advance to specfic requirements that an acceptable candidate has to fulfill. In this way, he can motivate $\mathcal{S}$ to restrict attention to candidates that are great for $\mathcal{R}$. However, if no candidate fulfilling these requirements is found, $\mathcal{R}$ is also committed to rejecting any other (possibly still medium-good) candidate, resulting in no utility for him (and $\mathcal{S}$).

More formally, in Delegation we have, similar to persuasion,

- Two agents called *sender* $\mathcal{S}$ and *receiver* $\mathcal{R}$
- n boxes. Box i contains a prize-pair (s_{ij}, r_{ij}), where $s_{ij} \geq 0$ is the prize for $\mathcal{S}$ and $r_{ij} \geq 0$ the prize for $\mathcal{R}$
- Set Θ_i of m_i possible prize-pair vectors $\theta_{ij} = (s_{ij}, r_{ij})$ for box i
- Joint distribution $\mathcal{D}$ over $\Theta = \Theta_1 \times \ldots \times \Theta_n$. $\mathcal{D}$ is known to both $\mathcal{S}$ and $\mathcal{R}$ apriori.

Now, in contrast to persuasion,

- $\mathcal{R}$ specifies a **decision scheme** $\psi : \bigcup_i \Theta_i \to \{0, 1\}$
- Nature draws the vector of prize-pairs in all boxes $\theta \sim \mathcal{D}$
- $\mathcal{S}$ sees θ. $\mathcal{R}$ does not see θ, knows only $\mathcal{D}$ and has committed to ψ
- $\mathcal{S}$ picks one box i and presents θ_{ij} to $\mathcal{R}$
- If $\psi(\theta_{ij}) = 1$, then $\mathcal{R}$ accepts, $\mathcal{S}$ and $\mathcal{R}$ receive s_{ij} and r_{ij}, resp.
- Otherwise, $\mathcal{R}$ rejects, and they both get nothing
- **Goal:** Find ψ^* that yields highest expected prize for $\mathcal{R}$.

Our main insight is an inherent connection to **prophet inequalities**. We describe it for Delegation with Independent Boxes, in which $\mathcal{D}_i$ is independent for each box $i \in [m]$. Algorithm 14 computes an approximate scheme, which not only approximates the optimal expected value for $\mathcal{R}$ achievable in the delegation scenario. It even approximates the expected optimal value that $\mathcal{R}$ would **obtain when searching though all boxes herself**.

Algorithm 14: Median-Prophets for DELEGATION WITH INDEPENDENT BOXES

Compute $r^* \leftarrow \mathbb{E}[\max_i r_{ij}]$, the expected maximal prize for $\mathcal{R}$
Compute the median τ with $p_0 = \Pr[r^* \geq \tau] \geq 1/2$ and $p_1 = \Pr[r^* > \tau] \leq 1/2$
Set $R_0 = \{r_{ij} \mid r_{ij} \geq \tau\}$ and $R_1 = \{r_{ij} \mid r_{ij} > \tau\}$
Compute q such that $q \cdot p_0 + (1-q) \cdot p_1 = 1/2$
With prob. q set $R \leftarrow R_0$; else $R \leftarrow R_1$
for *all* $i \in [n]$, $j \in [m_i]$ **do** $\psi(\theta_{ij}) \leftarrow \begin{cases} 1 & r_{ij} \in R \\ 0 & \text{otherwise} \end{cases}$
return ψ

Theorem 27. *Algorithm 14 computes an decision scheme for* DELEGATION WITH INDEPENDENT BOXES *that gives a 2-approximation of the expected optimal value in any box for* $\mathcal{R}$.

Proof. Algorithm 14 is a variant of Algorithm 4. It accepts any realization that is at least (in set R_0) or strictly more (in set $R_1 \subset R_0$) than a threshold τ. The threshold τ is not $r^*/2$ – instead, more in the spirit of Algorithm 5, we pick the "1/2-quantile" or "median" of the joint distribution for the optimal value r^*.

Facing the scheme ψ computed by Algorithm 14, $\mathcal{S}$ chooses a box with highest value s_{ij} among all boxes i with $\psi(\theta_{ij}) = 1$. Note that the exact same choice would emerge from the following **simulation**:

- $\mathcal{S}$ looks into all boxes and orders them in non-increasing order of s_{ij}
- $\mathcal{R}$ opens the boxes online in that order
- $\mathcal{R}$ accepts the first one for which $\psi(\theta_{ij}) = 1$.

The simulation is a PROPHET problem: $\mathcal{R}$ uses a threshold of at least τ or strictly more than τ for accepting the prize, as determined by Algorithm 14. $\mathcal{S}$ determines the arrival order of boxes. We will bound the competitive ratio by 2, **even if the arrival order** is determined by $\mathcal{S}$ **based on the realized prizes in the boxes**! Hence, $\mathcal{R}$ obtains at least half of the expected optimal value in any box, which proves the theorem.

First, consider r^*. If $r^* \geq \tau$, then $r^* = \max_{i\in[n]} r_{ij} = \tau + \max_{i\in[n]}(r_{ij} - \tau)$. Otherwise, $r^* \leq \tau = \tau + \max_{i\in[n]} 0$. Hence, overall,

$$\begin{aligned} r^* &\leq \tau + \mathbb{E}[\max_{i\in[n]} \max\{r_{ij} - \tau, 0\}] \leq \tau + \mathbb{E}\left[\sum_{i\in[n]} \max\{r_{ij} - \tau, 0\}\right] \\ &= \tau + \sum_{i\in[n]} \mathbb{E}[\max\{r_{ij} - \tau, 0\}] \end{aligned}$$

This upper bound for r^* has two parts – the threshold τ and a bonus term for each box. In analysis for the algorithm below, we approximate each part irrespective of the arrival order.

First, suppose there is at least one acceptable realization. Then a reward of at least τ is obtained irrespective of the arrival order. Turns out this happens with prob. at least 1/2.

For the bonus term of box i, we consider the case that no box $j \neq i$ has an acceptable realization. Then the algorithm picks the realization of box i if and only if it is acceptable, irrespective of the arrival order. Hence, it gets the bonus term. Turns out this happens with prob. at least $1/2$.

More formally, consider r_{alg}, the random prize value obtained by the algorithm.

- Y_i indicator random variable for the event that $\{r_{1j}, \ldots, r_{nj}\} \cap R = \{r_{ij}\}$, i.e., box i is the only acceptable one. Then $r_{alg} = r_{ij}$.
- All these events are distinct.
- $r_{alg} = 0$ when all $r_{ij} \notin R$. Otherwise, $r_{alg} \geq \tau$. We count the bonus above τ only in case there is *exactly one* $r_{ij} \in R$ (i.e., when $Y_i = 1$ for exactly one i).

$$\begin{aligned}\mathbb{E}[r_{alg}] &\geq \Pr[r_{alg} \in R] \cdot \tau + \mathbb{E}\left[\sum_{i\in[n]} Y_i(r_{ij} - \tau)\right] \\ &= \Pr[r_{alg} \in R] \cdot \tau + \sum_{i\in[n]} \mathbb{E}[Y_i(r_{ij} - \tau)]\end{aligned}$$

When $r^* \in R$, the algorithm also obtains a value in R for any choice of R_0 or R_1. Hence, $\Pr[r_{alg} \in R] = \Pr[r^* \in R]$. We choose $R = R_0$ with probability q and $R = R_1$ otherwise, and $p_i = \Pr[r^* \in R_i]$ for $i = 0, 1$ by definition. This shows

$$\Pr[r_{alg} \in R] = \Pr[r^* \in R] = q \cdot p_0 + (1 - q)p_1 = \frac{1}{2}$$

This implies

$$\begin{aligned}\mathbb{E}[r_{alg}] &\geq \Pr[r_{alg} \in R] \cdot \tau + \sum_{i\in[n]} \mathbb{E}[Y_i(r_{ij} - \tau)] \\ &= \frac{1}{2} \cdot \tau + \sum_{i\in[n]} \mathbb{E}[Y_i(r_{ij} - \tau)] \\ &= \frac{1}{2} \cdot \tau + \sum_{i\in[n]} \Pr[Y_i = 1] \cdot \mathbb{E}[r_{ij} - \tau \mid Y_i = 1]\end{aligned}$$

Due to independence among all boxes, we see that $\mathbb{E}[r_{ij} - \tau \mid Y_i = 1] = \mathbb{E}[r_{ij} - \tau \mid r_{ij} \in R]$, since box i is drawn independently. Now we observe for all $i \in [n]$

$$\begin{aligned}&\Pr[Y_i = 1] \cdot \mathbb{E}[r_{ij} - \tau \mid Y_i = 1] \\ &= \Pr[Y_i = 1] \cdot \mathbb{E}[r_{ij} - \tau \mid r_{ij} \in R] \\ &= \Pr[r_{ij} \in R] \cdot \prod_{k\neq i} \Pr[r_{kj} \notin R] \cdot \mathbb{E}[r_{ij} - \tau \mid r_{ij} \in R] \\ &= \prod_{k\neq i} \Pr[r_{kj} \notin R] \cdot \mathbb{E}[\max\{r_{ij} - \tau, 0\}] && r_{ij} > \tau \text{ implies } r_{ij} \in R_0 \cap R_1 \subseteq R \\ &\geq \prod_{k\in[n]} \Pr[r_{kj} \notin R] \cdot \mathbb{E}[\max\{r_{ij} - \tau, 0\}]\end{aligned}$$

$$= \Pr[r^* \notin R] \cdot \mathbb{E}[\max\{r_{ij} - \tau, 0\}]$$
$$= \frac{1}{2} \cdot \mathbb{E}[\max\{r_{ij} - \tau, 0\}] \qquad \text{since } \Pr[r^* \notin R] = 1 - \Pr[r^* \in R] = \frac{1}{2}$$

Hence,

$$\mathbb{E}[r_{alg}] \geq \frac{1}{2} \cdot \left(\tau + \sum_{i \in [n]} \mathbb{E}[\max(r_{ij} - \tau, 0)] \right) \geq \frac{1}{2} \cdot r^*.$$

□

Chapter 6

Stochastic Multi-Armed Bandits

6.1 Infinite Markov Decision Processes

Recall Section 3.2 and Markov decision processes with *finite time horizon*. A natural extension are MDPs with infinite time horizon. These eternal processes have important applications in machine learning, especially in the area of reinforcement learning.

The setup is similar: Sets $\mathcal{S}$ and $\mathcal{A}$ of states and actions, upon action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$, the process yields reward $r_a(s)$. It advances to state $s' \in \mathcal{S}$ with probability $p_a(s, s')$. Policy π considers in each step t the history[1] $s_0, \ldots, s_{t-1}$ and picks an action $\pi(s_0, \ldots, s_{t-1}) \in \mathcal{A}$. In this way, random sequences of states $s_0^\pi, \ldots, s_t^\pi$ and actions $a_0^\pi, \ldots, a_t^\pi$ evolve.

In infinite MDPs we study **time-discounted payoffs**, where future payoffs are less important than present ones. Formally, given a discount factor $\gamma \in (0, 1)$, the reward of policy π starting in state s_0 is

$$V(\pi, s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \cdot r_{a_t^\pi}(s_t^\pi)\right].$$

For interpretation, suppose, e.g., that the process terminates in step t independently with probability γ, and continues to the next step otherwise. Then $V(\pi, s_0)$ represents the expected payoff from playing the policy π.

Consider an optimal policy π^*. The following can be shown similarly as for finite-time MDPs:

- There is a **Markovian** π^*, in which $\pi^*(s_0, \ldots, s_{t-1}) = \pi^*(s_{t-1})$, i.e., the action depends only on the current state.
- A Markovian π^* chooses in each step t an action that maximizes the current reward plus the (discounted) future rewards from π^* starting from the next (random) state s' for the remaining (infinite) time horizon. Hence, the expected reward is

$$V^*(s) = V(\pi^*, s) = \max_{a \in \mathcal{A}} \left(r_a(s) + \gamma \cdot \sum_{s' \in \mathcal{S}} p_a(s, s') \cdot V(\pi^*, s') \right), \tag{6.1}$$

Eq. (6.1) is called *Bellman equation* and is a classic result in dynamic programming.

[1] Note that we start the process in round 0 here.

Computing the optimal policy by backwards induction is impossible due to the infinite time horizon. Instead, we can use linear programming to compute the values $v_s = V^*(s)$:

$$\begin{aligned} \text{Minimize} \quad & \sum_{s\in\mathcal{S}} v_s \\ \text{subject to} \quad & v_s \;\geq\; r_a(s) + \gamma \sum_{s'\in\mathcal{S}} p_a(s,s') \cdot v_{s'} \quad \text{for all } s \in \mathcal{S}, a \in \mathcal{A}. \end{aligned} \tag{6.2}$$

The $|\mathcal{A}|$ constraints for v_s express the max-operator in (6.1). Increasing values $v_{s'}$ only leads to further increase in the right-hand-sides of all constraints for v_s. However, the objective is to *minimize* the sum of all v_s. Thus, in an optimal solution, v_s gets raised as little as possible. As such, there must be a tight constraint for every v_s, which indicates an optimal action for π^* in s.

For many interesting MDPs, solving LP (6.2) is too time-consuming. Rather, we consider two simpler iterative processes that lead to an optimal policy, namely **value iteration** and **policy iteration**.

Value Iteration. Value iteration just iterates application of the Bellman equation (6.1). Start with any vector $v^{(0)} = (v_s^{(0)})_{s\in\mathcal{S}}$, and update $v_s^{(k+1)} = t(v^{(k)})_s$ in each step k for all $s \in \mathcal{S}$, using

$$t(v)_s = \max_{a\in\mathcal{A}} \left(r_s(a) + \gamma \sum_{s'\in\mathcal{S}} p_a(s,s') \cdot v_{s'} \right) \quad .$$

Clearly, if $v_s = V^*(s)$, then $t(v) = v$ is a fixed point of the iteration. We show that this is the *only* fixed point, and the iteration approaches it.

Theorem 28. *Value iteration converges to the unique fixed point* $\lim_{k\to\infty} v_s^{(k)} = V^*(s)$ *for all* $s \in \mathcal{S}$.

Proof. We first show that $t(v)$ is a **contraction**. Consider two vectors $v = (v_s)_{s\in\mathcal{S}}$ and $v' = (v'_s)_{s\in\mathcal{S}}$. After applying one step of the iteration, the maximum difference in the entries of any state $s \in \mathcal{S}$ shrinks by at least a factor γ.

- We define the distance of two vectors v and v' by

$$d(v,v') = \max_{s\in\mathcal{S}} |v_s - v'_s| = \|v - v'\|_\infty$$

- Consider v. Let a^* be an action that attains the maximum when computing $t(v)_s$. Note that a^* might not be an optimal action in $t(v')_s$, so

$$\begin{aligned} t(v_s) &= r_s(a^*) + \gamma \sum_{s'\in\mathcal{S}} p_{a^*}(s,s') \cdot v_{s'} \qquad \text{and} \\ t(v'_s) &\geq r_s(a^*) + \gamma \sum_{s'\in\mathcal{S}} p_{a^*}(s,s') \cdot v'_{s'}. \end{aligned}$$

- This implies

$$\begin{aligned} t(v_s) - t(v'_s) &\leq \gamma \sum_{s' \in \mathcal{S}} p_{a^*}(s, s') \cdot (v_{s'} - v'_{s'}) \\ &\leq \gamma \sum_{s' \in \mathcal{S}} p_{a^*}(s, s') \cdot d(v, v') \qquad \text{since } v_{s'} - v'_{s'} \leq \max_{s \in \mathcal{S}} |v_s - v'_s| = d(v, v') \\ &= \gamma \cdot d(v, v') \qquad \text{since } \sum_{s' \in \mathcal{S}} p_{a^*}(s, s') = 1 \end{aligned}$$

- The same holds if we reverse the roles of v and v'. Thus, $|t(v_s) - t(v'_s)| \leq \gamma \cdot d(v, v')$ for all $s \in \mathcal{S}$, and

$$\max_{s \in \mathcal{S}} |t(v)_s - t(v')_s| = d(t(v), t(v')) \leq \gamma \cdot d(v, v') \ .$$

Uniqueness: Suppose there are two different fixed points v^* and v^{**} of $t(v)$. Then $v^* = t(v^*)$ and $v^{**} = t(v^{**})$, so $d(v^*, v^{**}) = d(t(v^*), t(v^{**})) \leq \gamma \cdot d(v^*, v^{**})$ – a contradiction since $\gamma < 1$. Hence, there is a unique fixed point of $t(v)$.

Convergence: Consider the fixed point v^* and our iteration $v^{(k)}$. The distance to the fixed point shrinks every step

$$d(v^{(k+1)}, v^*) = d(t(v^{(k)}), t(v^*) \leq \gamma \cdot d(v^{(k)}, v^*) \leq \gamma^{k+1} \cdot d(v^{(0)}, v^*).$$

The initial distance $d(v^{(0)}, v^*)$ is finite and independent of t, so the process must converge. □

Policy Iteration. Here we start with any Markovian policy $\pi^{(0)}$ and iteratively improve it until it stops changing. We improve a policy $\pi^{(k)}$ as follows: First compute all values $V(\pi^{(k)}, s)$ by solving the system of linear equations

$$V(\pi^{(k)}, s) = r_{\pi^{(k)}(s)}(s) + \gamma \cdot \sum_{s' \in \mathcal{S}} p_{\pi^{(k)}(s)}(s, s') \cdot V(\pi^{(k)}, s') \qquad \text{for all } s \in \mathcal{S}.$$

Now for each state $s \in \mathcal{S}$, compute the action $a \in \mathcal{A}$ that maximizes the reward $r_a(s) + \gamma \cdot \sum_{s' \in \mathcal{S}} p_a(s, s') \cdot V(\pi^{(k)}, s')$ and set $\pi^{(k+1)}(s) = a$.

Theorem 29. *Policy iteration converges to an optimal policy in a finite number of steps.*

Proof. Any fixed-point policy of the iteration with $\pi^{(k+1)} = \pi^{(k)}$ fulfills the Bellman equation (6.1) and thus is optimal. To show that policy iteration converges, note that there are only $|\mathcal{S}|^{|\mathcal{A}|}$ Markovian policies. Hence, any non-converging policy iteration must cycle. We will show that no cycle is possible.

A cycle is absent if $V(\pi^{(k+1)}, s) \geq V(\pi^{(k)}, s)$ for all $s \in \mathcal{S}$ and all $k \geq 0$.

- Fix a step k. Consider an auxiliary sequence of policies $\pi'_0, \pi'_1, \ldots$
- In π'_i we follow $\pi^{(k+1)}$ for the first i steps and then switch to using $\pi^{(k)}$.
- Then, in particular, $V(\pi^{(k)}, s) = V(\pi'_0, s)$ and $V(\pi^{(k+1)}, s) = \lim_{i \to \infty} V(\pi'_i, s)$.

- Suppose
$$V(\pi'_i, s) \geq V(\pi'_{i-1}, s) \qquad \text{for all } s \in \mathcal{S}, i \in \mathbb{N} \tag{6.3}$$
then $V(\pi^{(k+1)}, s) \geq V(\pi^{(k)}, s)$ for all $s \in \mathcal{S}$ and policy iteration does not cycle.

We show (6.3) by induction, this implies convergence and the theorem.

- Base case:
$$\begin{aligned} V(\pi'_0, s) &= r_{\pi^{(k)}(s)}(s) + \gamma \cdot \sum_{s' \in \mathcal{S}} p_{\pi^{(k)}(s)}(s, s') \cdot V(\pi^{(k)}, s') \\ V(\pi'_1, s) &= r_{\pi^{(k+1)}(s)}(s) + \gamma \cdot \sum_{s' \in \mathcal{S}} p_{\pi^{(k+1)}(s)}(s, s') \cdot V(\pi^{(k)}, s') \end{aligned} \tag{6.4}$$
since π'_0 behaves entirely as $\pi^{(k)}$, whereas π'_1 makes the *first* choice with $\pi^{(k+1)}$ and then behaves as $\pi^{(k)}$.
- By definition of policy iteration, $\pi^{(t+1)}(s)$ is the action that maximizes the right-hand side(s) of (6.4). Thus, $V(\pi'_1, s) \geq V(\pi'_0, s)$, which proves the base case.
- Induction Step: Observe that
$$\begin{aligned} V(\pi'_{i-1}, s) &= r_{\pi^{(k+1)}(s)}(s) + \gamma \cdot \sum_{s' \in \mathcal{S}} p_{\pi^{(k+1)}(s)}(s, s') \cdot V(\pi'_{i-2}, s') \\ V(\pi'_i, s) &= r_{\pi^{(k+1)}(s)}(s) + \gamma \cdot \sum_{s' \in \mathcal{S}} p_{\pi^{(k+1)}(s)}(s, s') \cdot V(\pi'_{i-1}, s') \ . \end{aligned}$$
The hypothesis $V(\pi'_{i-1}, s) \geq V(\pi'_{i-2}, s)$ now directly implies $V(\pi'_i, s) \geq V(\pi'_{i-1}, s)$, since the right-hand sides of the lower equation are pointwise larger for all $s' \in \mathcal{S}$.

□

6.2 Markovian Multi-Armed Bandits

Multi-armed bandit problems play a central role in machine learning, especially in the area of online learning. We start with a variant that represents an MDP with infinite time horizon. The problem has a huge state space, so the techniques from the previous section are not efficiently applicable.

Consider the Markovian Single-Armed Bandit problem:

- MDP with two actions $\mathcal{A} = \{\texttt{play}, \texttt{pause}\}$.
- `play`: Probability distributions and rewards are arbitrary.
- `pause`: Always "pause" at the current state, i.e., $p_{\texttt{pause}}(s, s) = 1$ and $r_{\texttt{pause}}(s) = 0$.

Example 8. You are a gambler in a stylized lottery. Every week you can decide to play or to skip the round. If you play, you must invest 1 unit of money, and you win with 1% probability. Once you have won, you can obtain a payment of 5 units of money every week, until the lottery terminates. Every week the lottery terminates with a probability $1 - \gamma$.

The states are $\mathcal{S} = \{s, t\}$, where t means you have won. For s, we have $p_{\texttt{play}}(s, s) = 0.99$, $p_{\texttt{play}}(s, t) = 0.01$, and reward $r_{\texttt{play}}(s) = -1$. For t we have $p_{\texttt{play}}(t, t) = 1$ and $r_{\texttt{play}}(t) = 5$. We derive an optimal Markovian policy π^*.

[Pic: States, Rewards, Transitions]

In state t, the discounted reward from playing is $r_t = \sum_{i=0}^{\infty} \gamma^i \cdot 1 \cdot 5 = \frac{5}{1-\gamma} > 0$. Hence, $\pi^*(t) = \texttt{play}$ always. In state s, `pause` is the best action if and only if the discounted reward for `play` is $r_s \leq 0$. Note

$$r_s = -1 + \gamma \cdot (0.99 \cdot r_s + 0.01 \cdot r_t) = -1 + \gamma \cdot \left(0.99 \cdot r_s + \frac{0.05}{1-\gamma}\right),$$

which implies

$$r_s(1 - 0.99\gamma) = \frac{0.05\gamma}{1-\gamma} - 1.$$

Since $1 - 0.99\gamma > 0$, we have $r_s \leq 0$ if and only if $\frac{0.05\gamma}{1-\gamma} \leq 1$. Hence, $\pi^*(s) = \texttt{pause}$ when $0.05\gamma \leq 1 - \gamma$, i.e., $\gamma \leq 1/1.05 \approx 0.9523$. Otherwise, $\pi^*(s) = \texttt{play}$. Notably, the expected duration of the lottery in that case is $\sum_{i=0}^{\infty} \gamma^i(1-\gamma) \cdot i = \frac{\gamma}{1-\gamma} > \frac{1/1.05}{1-1/1.05} = 20$ weeks. ■

Now consider the MARKOVIAN MULTI-ARMED BANDIT problem

- Parallel composition of single-armed bandits.
- States $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ are vectors of states for the n single-armed bandits.
- Actions $\mathcal{A} = \{\texttt{play}_1, \ldots, \texttt{play}_n, \texttt{pause}\}$; action $\texttt{play}_i$ means `play` on the i-th single-armed bandit and `pause` on all others. `pause` pauses all arms.
- Arms operate independently, but we can advance only one arm at a time.

If a Markovian policy decides to `pause`, it remains in that state and therefore keeps pausing eternally. For $\gamma = 1$ it would be irrelevant in which order we play the arms. Since $\gamma < 1$, we want high rewards on the chosen arms as early as possible, and the order is important!

Example 9. It seems intuitive to greedily pick an arm with highest upcoming reward. However, this is not optimal. Consider two arms with $\mathcal{S}_1 = \{s_1, s_2, s_3\}$ and $\mathcal{S}_2 = \{t_1, t_2\}$. Initially the arms are in states s_1 and t_2.

For arm 1, we have no initial reward: $r_{\texttt{play}}(s_1) = 0$ and $p_{\texttt{play}}(s_1, s_2) = 1/2$, $p_{\texttt{play}}(s_1, s_3) = 1/2$. If we move to state s_2, there is a big reward: $r_{\texttt{play}}(s_2) = 1000$ and $p_{\texttt{play}}(s_2, s_3) = 1$. In state s_3, there is no reward: $r_{\texttt{play}}(s_3) = 0$ and $p_{\texttt{play}}(s_3) = 0$.

For arm 2, we have a bigger initial reward: $r_{\texttt{play}}(t_1) = 10$ and $p_{\texttt{play}}(t_1, t_2) = 1$, but then no later reward: $r_{\texttt{play}}(t_2) = 0$ and $p_{\texttt{play}}(t_2, t_2) = 1$.

[Pic: States, Rewards, Transitions]

If we first play arm 1, this gives no immediate reward, but with 50% probability enables us to reap a reward of 1000 in the next step. Clearly, one would want to exploit this option as early as possible. Instead, arm 2 gives reward of 10 the first time it is played. For large γ, this should be done only *after* the expected reward of 500 in arm 1 is secured. ■

Towards an optimal policy, we consider a separate, auxiliary single-armed bandit problem for each arm $i \in [n]$. In this auxiliary problem,...

- Every time you `play`, you have to pay λ.

- The maximum reward for an optimal policy in the auxiliary problem for arm i is

$$V_i(s,\lambda) = \max\left\{0, r_{\texttt{play},i}(s) - \lambda + \gamma \cdot \sum_{s' \in \mathcal{S}} p_{\texttt{play},i}(s,s') \cdot V_i(s',\lambda)\right\}$$

- For increasing λ, the value $V_i(s,\lambda)$ continuously decreases.

$\Rightarrow$ There is a largest value $\delta(s)$ such that $V_i(s,\delta(s)) = 0$ and `play` is as good as `pause` in the auxiliary problem for arm i. Formally

$$\delta_i(s) = \sup\{\lambda \mid V_i(s,\lambda) > 0\} = \inf\{\lambda \mid V_i(s,\lambda) = 0\}$$

- $\delta_i(s)$ is called the **fair charge** or **Gittins index** of arm i in state s.

Example 10. Consider a single arm with states $\{A, B, C, D\}$, initially in state A. The transitions are deterministic $A \to B \to C \to D \to D \to D \to \ldots$. The rewards are $r_{\texttt{play}}(A) = 1$, $r_{\texttt{play}}(B) = 12$, $r_{\texttt{play}}(C) = 1$, and $r_{\texttt{play}}(D) = 0$. Assume $\gamma = 2/3$.

The fair charge of arm D is 0. For arm C, the fair charge is 1 – then it is optimal to `play` in C and `pause` in D. For B the fair charge is 12 – then it is optimal to `play` in B and `pause` in C and D.

Finally, suppose it is optimal to `play` in A. Then suppose we `play` in B and `pause` in C and D. In this case, we obtain a reward of

$$1 - \lambda + \frac{2}{3} \cdot (12 - \lambda) = 9 - \frac{5}{3} \cdot \lambda$$

and the fair charge fulfills $9 = \frac{5}{3}\lambda$, i.e., $\lambda = 27/5 = 5.4 > 1$. Now with this value of λ, it is indeed optimal to `play` in B and `pause` in C and D. ■

Consider the **Gittins-Index policy** π_G: If there is an arm with a positive Gittins index, choose $\texttt{play}_i$ for an arm i with the highest Gittins index. Otherwise, play `pause`.

The main result of this section is the following theorem.

Theorem 30. *π_G is an optimal policy for* Markovian Multi-Armed Bandit.

We first prove the theorem for the Markovian Single-Armed Bandit problem. Consider the auxiliary problem with charge λ. An optimal policy for the auxiliary problem can be derived from the fair charges $\delta(s)$: Pick `play` if $\delta(s) > \lambda$ and `pause` if $\delta(s) < \lambda$. For $\delta(s) = \lambda$, both are optimal. Now based on this, let us consider the Markovian Single-Armed Bandit problem without charges.

Lemma 19. *Consider a policy for a single arm that first only chooses* ***play*** *and then only chooses* ***pause****. Let τ be the step in which it chooses* ***pause*** *for the first time. Then*

$$\mathbb{E}\left[\sum_{t=0}^{\tau-1} \gamma^t \cdot r_{\texttt{play}}(s_t)\right] \leq \mathbb{E}\left[\sum_{t=0}^{\tau-1} \gamma^t \cdot \min_{t' \leq t} \delta(s_{t'})\right] \tag{6.5}$$

with equality if $\delta(s_\tau) = \min_{t' \leq \tau} \delta(s_{t'})$ with probability 1.

Proof. First consider the case when $\delta(s_\tau) = \min_{t' \le \tau} \delta(s_{t'})$ with probability 1.

- Policy stops to `play` in some state that has smallest fair charge seen so far.
- An execution decomposes into **phases** of random length.
- Let $\tau_0 = 0$ and τ_{k+1} the first step $t \ge \tau_k$ where $\delta(s_t) < \delta(s_{\tau_k})$.
- The policy will `pause` at one of the time points τ_k.

[Pic: Fair charges over time, phases, time steps τ_k, stopping time]

An alternative interpretation:

- Fix everything until time τ_k.
- At time τ_k, start an optimal policy for the arm with charge $\lambda = \delta(\tau_k)$.
- Policy stops at time $t = \tau_{k+1}$, the first step in which the charge $\delta(s_t) < \lambda = \delta(\tau_k)$.
- Consider expected reward of optimal policy **in the auxiliary problem with charge** $\lambda = \delta(\tau_k)$ starting from τ_k. By definition of fair charge, the expected reward is 0. Also, the optimal policy plays in the time interval $\{\tau_k, \dots, \tau_{k+1} - 1\}$, so

$$\mathbb{E}\left[\sum_{t=\tau_k}^{\tau_{k+1}-1} \gamma^{t-\tau_k} \cdot (r_{\texttt{play}}(s_t) - \delta(s_{\tau_k})) \,\middle|\, \tau_k\right] = 0$$

- This implies for the (non-auxiliary) original problem

$$\begin{aligned}\mathbb{E}\left[\sum_{t=\tau_k}^{\tau_{k+1}-1} \gamma^{t-\tau_k} r_{\texttt{play}}(s_t) \,\middle|\, \tau_k\right] &= \mathbb{E}\left[\sum_{t=\tau_k}^{\tau_{k+1}-1} \gamma^{t-\tau_k} \delta(s_{\tau_k}) \,\middle|\, \tau_k\right] \\ &= \mathbb{E}\left[\sum_{t=\tau_k}^{\tau_{k+1}-1} \gamma^{t-\tau_k} \min_{t' \le t} \delta(s_{t'}) \,\middle|\, \tau_k\right] \quad .\end{aligned}$$

- The statement for $\delta(s_\tau) = \min_{t' \le \tau} \delta(s_{t'})$ follows by summing over all k.

Now consider a general policy. We only stop early in one of the phases (say, phase j). When we stop, the fair charge of the current state is larger than the charge of the phase $\lambda = \delta(s_{\tau_j})$. The reward in the auxiliary problem for phase j can be at most 0. This proves the inequality for general policies. □

Lemma 20. *Consider an arbitrary policy π for* Markovian Single-Armed Bandit. *We denote by T the set*[2] *of time steps where π picks* ***play****. Then*

$$\mathbb{E}\left[\sum_{t \in T} \gamma^t r_{\texttt{play}}(s_t)\right] \le \mathbb{E}\left[\sum_{t \in T} \gamma^t \min_{t' \le t} \delta(s_{t'})\right]$$

with equality if $\delta(s_t) = \min_{t' \le t} \delta(s_{t'})$ for all $t \notin T$ with probability 1.

Proof. Note that the previous lemma proves the result when $T = \{0, 1, \dots, \tau - 1\}$.

- We can apply the argument to the case $T = \{t', \dots, t' + \tau - 1\}$ for some t'.
- Then $\delta(s_0) = \dots = \delta(s_{t'})$, so both sides of (6.5) simply get multiplied by $\gamma^{t'}$.

[2]Note that T is usually a random set depending on realized states during the execution of π.

- Generally, T is a union of disjoint time intervals, each one has the form $\{t', \ldots, t'+\tau-1\}$.
- Apply the above argument to each interval, add up the resulting inequalities.
- Linearity of expectation implies the result.

□

Proof of Theorem 30. Let $T_i = \{t \mid \pi_G(s_t) = \texttt{play}_i\}$, i.e., the set of steps where π_G plays arm i. We observe how the Gittins index $\delta_i(s_t)$ of arm i changes over time.

- If $t \notin T_i$, then $\delta_i(s_{t+1}) = \delta_i(s_t)$. Otherwise, $\delta_i(s_{t+1})$ can differ from $\delta_i(s_t)$.
- Suppose we play arm i. We continue until δ_i falls below the value it had when we started playing arm i and also stops being the one with maximum index among all other arms.
- Thus, when we stop playing arm i, then δ_i is at an all-time low.
- This implies: If $t \notin T_i$ then $\delta_i(s_t) \leq \min_{t' \leq t} \delta_i(s_{t'})$.
- This property allows to apply Lemma 20. The expected reward from arm i is, thus,

$$\mathbb{E}\left[\sum_{t \in T_i} \gamma^t \min_{t' \leq t} \delta_i(s_{t'})\right]$$

- Overall, the expected reward[3] from π_G is

$$R(\pi_G) = \sum_{i \in [n]} \mathbb{E}\left[\sum_{t \in T_i} \gamma^t \min_{t' \leq t} \delta_i(s_{t'})\right] = \mathbb{E}\left[\sum_{i \in [n]} \sum_{t \in T_i} \gamma^t \min_{t' \leq t} \delta_i(s_{t'})\right]$$

For any other policy π', we can make similar observations. However, by Lemma 20 the expected reward of any policy π' is only *upper bounded* by $R(\pi')$.

The final step of the proof is to show that $R(\pi)$ is maximized for $\pi = \pi_G$.

- Consider π_G and any policy π.
- Assumption: Policies play every arm infinitely often, i.e., all $|T_i| = \infty$ in both policies.
- Proof for finite sets T_i also works, but much more messy!
- $x_t = \min_{t' \leq t} \delta_i(s_t)$ for the arm with $t \in T_i$ in π_G
- $y_t = \min_{t' \leq t} \delta_i(s_t)$ for the arm with $t \in T_i$ in π
- Arm i transitions randomly to another state in $\mathcal{S}_i$ if it is played.
- Fix coin flips and resulting transitions arbitrarily. We denote the transitions by ζ.
- Given ζ, now $x_0, x_1, \ldots$ and $y_0, y_1, \ldots$ are fixed.
- Each arm played infinitely often: The arms go through the same state transitions ζ. Only the order of arms in the sequences varies.
- Sequences x and y contain the same numbers. For π_G we have $x_0 \geq x_1 \geq x_2 \geq \ldots$, i.e., the sequence is non-increasing. This implies

$$R(\pi, \zeta) = \sum_{t=0}^{\infty} \gamma^t y_t \leq \sum_{t=0}^{\infty} \gamma^t x_t = R(\pi_G, \zeta) \ .$$

- Holds for any choice ζ of the random transitions of all the arms ⇒ holds in expectation.

□

[3] $R(\pi_G)$ depends on the division of time steps into sets T_i, which results from the action choices of π_G.

6.3 Stochastic Multi-Armed Bandits

In previous sections, we studied MDPs with explicit stochastic information about the behavior of the process. In this section, we move away from this assumption. We consider a stateless multi-armed bandit problem, where the reward of each arm is drawn independently from some distribution. The main challenge is that *we don't know these distributions in advance*. The goal is to quickly learn which arm is the best one.

In the Stochastic Multi-Armed Bandit problem

- There are n arms and T rounds. Each arm i has a distribution $\mathcal{D}_i$.
- We assume $T \geq n$, and each $\mathcal{D}_i$ maps to the unit interval $[0, 1]$.
- In each round $t = 1, \ldots, T$:
- Nature draws a reward $R_{it} \in [0, 1]$ for each arm i independently from $\mathcal{D}_i$.
- Distributions and all draws *remain unknown*. We choose an arm $i_t \in [n]$.
- We see and obtain reward $R_t = R_{i_t,t}$, but *do not see* rewards of arms $j \neq i_t$ in round t.

Let μ_i be the (initially unknown) expected reward of arm $i \in [n]$.

- If we knew the distributions, we would always pull the best arm $i^* = \arg\max_i \mu_i$.
- Instead, we must first learn about $\mathcal{D}_i$ and μ_i's by observing the random draws.
- We only get expected reward $\mathbb{E}\left[\sum_{t=1}^{T} R_t\right] \leq T \cdot \mu_{i^*}$.
- **Goal**: Maximize expected reward, or put differently, mininize the expected **regret**:

$$Regret(T) = T \cdot \mu_{i^*} - \sum_{t=1}^{T} \mathbb{E}[R_t]$$

Explore-and-Exploit A simple algorithm: First *exploration*, then *exploitation*!

- In round $(i-1) \cdot k + 1, \ldots, i \cdot k$: Sample arm i for k rounds, for each $i \in [n]$
- Let $\hat{\mu}_i$ be the average reward of arm i during this exploration phase.
- In rounds $kn + 1, \ldots, T$ play arm with largest $\hat{\mu}_i$.

Choosing a suitable value for k, we obtain a reasonably good estimate for each μ_i, without sacrificing too many rounds with potentially large regret during exploration. The resulting regret is sublinear in T when T grows large.

Theorem 31. *For $k = (T/n)^{2/3}$ the regret of the simple algorithm is at most $O(n^{1/3}T^{2/3}\ln(nT))$.*

The proof uses a Hoeffding inequality – a concentration result similar to the Chernoff bound, for bounded independent variables (not necessarily Bernoulli ones). Intuitively, when we have many bounded variables, their average is close to expectation with a very large probability.

Theorem 32. *Let $X_1, \ldots X_n$ be independent random variables with $a_i \leq X_i \leq b_i$ and $\bar{X} = \frac{1}{N}\sum_{i=1}^{n} X_i$ be the average. Then the following* Hoeffing inequality *holds:*

$$\Pr[|\bar{X} - \mathbb{E}[\bar{X}]| \geq \delta] \quad \leq \quad 2\mathsf{e}^{-\frac{2n^2\delta^2}{\sum_i b_i - a_i}} \qquad \text{for all } \delta \geq 0$$

The proof of Theorem 31 is a combination of Hoeffing and union bounds:

Algorithm 15: UCB1-Algorithm for Stochastic Multi-Armed Bandit

for *round* $t = 1, \ldots, n$ **do** pull arm t, set $S_t \leftarrow R_{tt}$ and $P_t^{(n)} \leftarrow 1$
for *round* $t = n+1, \ldots, n$ **do**
  Set $\hat{\mu}_i^{(t)} \leftarrow S_i / P_i^{(t)}$ and $cb_i^{(t)} \leftarrow \hat{\mu}_i^{(t)} + \sqrt{\frac{\ln T}{P_i}}$ for all $i \in [n]$
  Pull arm $i_t = \arg\max_i cb_i^{(t)}$
  Update $S_{i_t} \leftarrow S_{i_t} + R_{i_t,t}$
  Update $P_{i_t}^{(t+1)} \leftarrow P_{i_t}^{(t)} + 1$, and $P_i^{(t+1)} \leftarrow P_i^{(t)}$ for $i \neq i_t$

- Hoeffding: For each arm i after exploration, probability that $|\hat{\mu}_i - \mu_i| \geq \delta$ is exponentially small
- Union: Probability that $|\hat{\mu}_i - \mu_i| \geq \delta$ for *all* arms is (n times) exponentially small
- Hence, with large probability an arm with best $\hat{\mu}_i$ has $\mu_i \geq \mu_{i^*} - 2\delta$. Then regret is at most $kn \cdot 1$ in the exploration phase and at most $(T - kn) \cdot 2\delta$ in the exploitation phase. Otherwise, with small probability regret is at most T.
- Finally, optimize k and δ: If $k = (T/n)^{2/3}$ and

$$\delta = \sqrt{\frac{n^{2/3}}{2 \cdot T^{2/3}} \ln \frac{2 \cdot n^{2/3}}{T^{1/3}}}$$

  the resulting calculation of the expected regret yields the bound in the theorem.

The details are left as an exercise.

UCB Algorithm The previous approach is not sensitive enough to the evolution of rewards – if an arm is very bad, it could be disregarded much earlier. Algorithm 15 is the *UCB1 (upper confidence bound, version 1)* algorithm. In every round t, it uses previous observations to compute an empirical average $\hat{\mu}_i^{(t)}$, for each arm $i \in [n]$. The more often we pull an arm, the closer this is to μ_i. Based on $\hat{\mu}_i^{(t)}$ and P_i^t (number of times arm i was pulled *before round* t), we compute $cb_i^{(t)}$ – an estimation on μ_i via a confidence interval of $\sqrt{(\ln T)/P_i}$ around $\hat{\mu}_i^{(t)}$. UCB always chooses an arm with highest confidence bound.

Theorem 33. *The expected regret of the UCB1 algorithm is at most* $\sum_{i \neq i^*} \frac{4 \ln T}{\Delta_i} + 4\Delta_i$*, where* $\Delta_i = \mu_{i^*} - \mu_i$

Note that Δ_i is usually a constant independent of T. Thus, asymptotically for large T the regret becomes as small as $O(\log T)$. We are also interested in the behavior of the regret in n, which is only linear as long as Δ_i are constants independent of n.

The main step of the proof is the following lemma.

Lemma 21. *For every arm* $i \in [n]$*, we denote by* $P_i = P_i^{(T+1)}$ *the number of times that arm* i *is pulled overall. It holds that* $\mathbb{E}[P_i] \leq s_i + 4$*, where* $s_i = \frac{4 \ln T}{\Delta_i^2}$*.*

We prove the lemma below. First, let us observe how it can be used to prove the theorem.

Proof of Theorem 33. We define the following random variables for each round t.

- $X_{it} = 1$ if UCB1 picks arm i in round t, 0 otherwise.
- Recall R_{it} is reward drawn in round t for arm i (before the decision of the algorithm)
- X_{it} and R_{it} are independent – choice without seeing rewards in round t.
- Hence $\mathbb{E}[X_{it}R_{it}] = \mathbb{E}[X_{it}]\mathbb{E}[R_{it}] = \mathbb{E}[X_{it}]\mu_i$

Since $\mathbb{E}[R_t] = \mathbb{E}[\sum_{i=1}^n X_{it}R_{it}]$, we have by linearity of expectation

$$\mathbb{E}\left[\sum_{t=1}^T R_t\right] = \mathbb{E}\left[\sum_{t=1}^T\sum_{i=1}^n X_{it}R_{it}\right] = \sum_{i=1}^n\sum_{t=1}^T \mathbb{E}\left[X_{it}\right]\mu_i = \sum_{i=1}^n \mathbb{E}\left[\sum_{t=1}^T X_{it}\right]\mu_i = \sum_{i=1}^n \mathbb{E}[P_i]\mu_i.$$

Note that $\sum_i P_i = T$ always, so the regret

$$T\mu_{i^*} - \mathbb{E}\left[\sum_{t=1}^T R_t\right] = \sum_{i=1}^n \mathbb{E}[P_i](\mu_{i^*} - \mu_i) = \sum_{i=1}^n \mathbb{E}[P_i]\Delta_i,$$

and the theorem follows by applying the upper bound on $\mathbb{E}[P_i]$ from Lemma 21. □

It remains to prove Lemma 21. Notably, it concerns only a single arm i and says that the larger Δ_i the earlier UCB1 stops pulling that arm. The main argument is encapsulated in the following lemma. It relates the precision of the estimation in $\hat{\mu}_i^{(t)}$ to $P_i^{(t)}$, i.e., the number of times an arm has been chosen.

Lemma 22. *For every $i \in [n]$ we have*

$$\Pr\left[\exists t : |\hat{\mu}_i(t) - \mu_i| \geq \sqrt{\frac{\ln T}{P_i^{(t)}}}\right] \leq \frac{2}{T} \ .$$

Proof. We cannot apply the Hoeffding inequality directly to $\hat{\mu}_i^{(t)}$ for a fixed t, since both $\hat{\mu}_i^{(t)}$ and the root-term depend on the random number $P_i^{(t)}$ of times we pulled arm i so far. Let us instead consider the event that we pull arm i for the k-th time.

- The decision to pull arm i in round t depends only on previous draws of this and other arms. Reward R_{it}, however, is *independent* of all previous draws.
- Let t_k be the (random) round, in which we pull arm i for the k-th time.

$\Rightarrow$ $\mu_i^{(t_k+1)} = \sum_{j=1}^k R_{i,t_j}/k$, a sum of independent random variables divided by $P_i^{(t_k+1)} = k$.

- Now observe that $\exists t : |\hat{\mu}_i^{(t)} - \mu_i| \geq \sqrt{\frac{\ln T}{P_i^{(t)}}} \iff \exists k : |\hat{\mu}_i^{(t_k+1)} - \mu_i| \geq \sqrt{\frac{\ln T}{k}}$.
- For a fixed k and the rewards $R_{i,t_1}, \ldots, R_{i,t_k}$ we can apply Hoeffdings inequality:

$$\Pr\left[\left|\frac{1}{k}\sum_{j=1}^k R_{i,t_j} - \mu_i\right| \geq \sqrt{\frac{\ln T}{k}}\right] \leq 2\mathsf{e}^{-\frac{2k^2(\ln T)/k}{k}} = \frac{2}{T^2} \ .$$

- This defined at most T events, one for each k. A union bound implies

$$\Pr\left[\exists k : \left|\frac{1}{k}\sum_{j=1}^k R_{i,t_j} - \mu_i\right| \geq \sqrt{\frac{\ln T}{k}}\right] \leq \sum_{t=1}^T \frac{2}{T^2} = \frac{2}{T} \ .$$

□

The lemma shows that $cb_i^{(t)}$ is close to μ_i with large probability – it's unlikely that $cb_i^{(t)}$ is much larger or smaller than μ_i.

Corollary 2. *For every $i \in [n]$ we have*

$$\Pr\left[\exists t : cb_i^{(t)} \leq \mu_i\right] \leq \frac{2}{T} \qquad \textit{and} \qquad \Pr\left[\exists t : cb_i^{(t)} \geq \mu_i + 2\sqrt{\frac{\ln T}{P_i^{(t)}}}\right] \leq \frac{2}{T} \ .$$

Using this insight, we prove Lemma 21, by which we complete the proof of Theorem 33.

Proof of Lemma 21. We first analyze the case when $P_i > s_i$, i.e., arm i is pulled often.

- $P_i > s_i \Rightarrow$ there is round t' with $P_i^{(t')} = s_i$ and i chosen again.
- i chosen because $cb_i^{(t')}$ is maximal, so $cb_i^{(t')} \geq cb_{i^*}^{(t')}$.
- We apply Corollary 2 to i^* and i:

$$\Pr\left[\exists t : cb_{i^*}^{(t)} \leq \mu_{i^*}\right] \leq \frac{2}{T} \qquad \text{and} \qquad \Pr\left[\exists t : cb_i^{(t)} \geq \mu_i + 2\sqrt{\frac{\ln T}{P_i^{(t)}}}\right] \leq \frac{2}{T} \ . \quad (6.6)$$

- Union bound: Probability at least one of the two happens is at most $4/T$.
- Thus, neither one happens with probability at least $1 - 4/T$. Then for all rounds t, we have $cb_{i^*}^{(t)} > \mu_{i^*}$ and $cb_i^{(t)} < \mu_i + 2\sqrt{\frac{\ln T}{P_i^{(t)}}}$. Hence, for round t' with $P_i^{(t')} = s_i$

$$cb_{i^*}^{(t')} > \mu_{i^*} \qquad \text{and} \qquad cb_i^{(t')} < \mu_i + 2\sqrt{\frac{\ln T}{s_i}} = \mu_i + \Delta_i = \mu_{i^*} \ ,$$

  which implies a contradiction to $cb_i^{(t')} \geq cb_{i^*}^{(t')}$.

$\Rightarrow$ If $P_i > s_i$, at least one of the events in (6.6) **must occur**, and so $\Pr[P_i > s_i] \leq 4/T$.

Finally, observe the following upper bound

$$\begin{aligned}\mathbb{E}[P_i] &\leq \Pr[P_i \leq s_i] \cdot s_i + \Pr[P_i > s_i] \cdot T \\ &\leq s_i + \Pr[P_i > s_i] \cdot T \\ &\leq s_i + 4.\end{aligned}$$

□

Chapter 7

Adversarial No-Regret Learning

7.1 Majority Algorithms and the Experts Problem

In the previous chapter, we have considered stochastic versions of the multi-armed bandit problem. Here we advance to a non-stochastic model, in which rewards for each arm are determined by an unknown entity. In the worst case, there might even be an adversary that fixes the rewards in a way to make our policy achieve as little reward as possible.

We start with an ExpertClassification problem, in which

- T items (say, pictures) arrive sequentially over time
- n classifiers inspect all pictures and issue their evaluation
- Each round t, each classifier tells us whether she believes the picture contains a cat ($x_i^{(t)} = 1$) or not ($x_i^{(t)} = 0$)
- Based on these numbers, we also decide $y^{(t)} \in \{0, 1\}$. Then we get feedback if the picture really contains a cat, i.e., we see if we and which of classifiers made a mistake.
- **Goal:** Small number of mistakes, comparable to the **best classifier in hindsight**

Since the classifiers might not be perfect, they make mistakes every once in a while. We only have access to their evaluations, so our goal is to be as good as the best one in hindsight.

Our approach is a simple *Weighted Majority* algorithm (Algorithm 16). It assigns each classifier i a weight $w_i^{(t)} \in (0, 1]$. Starting with $w_i^{(1)} = 1$, the weight is decreased by a factor $(1 - \eta)$ every time the classifier makes a mistake. The parameter $0 < \eta \leq 1/2$ is called a *learning rate*. The majority algorithm simply follows a weighted majority vote – if the weighted average prefers 1 (i.e., $W_H^{(t)} \geq W_L^{(t)}$), then $y^{(t)} = 1$, and 0 otherwise.

Intuitively, over time we have highest weight on the classifiers that made the least number of mistakes. Hence, their vote has a larger impact than the one of classifiers that have been wrong a lot in the past. The role of η is to steer this adaptation – the larger η, the more aggressively we punish single-round mistakes and follow classifiers that have been correct recently. For small η, we target the longer time horizon, making more gentle adjustments, but thereby also listening to the evaluation of bad classifiers for a longer time.

Theorem 34. *Let $M_i^{(T)}$ be the total number of mistakes of classifier i in T rounds. Weighted*

Algorithm 16: Weighted Majority for ExpertClassification

Set $w_i^1 = 1$ for all $i \in [n]$
for *round* $t = 1, \dots, T$ **do**
  Observe numbers $x_i^{(t)}$ for all $i \in [n]$
  Set $W^{(t)} = \sum_{i\in[n]} w_i^{(t)}$ and $W_H^{(t)} = \sum_{i:x_i^{(t)}=1} w_i^{(t)}$
  if $W_H^{(t)} \geq W^{(t)} - W_H^{(t)}$ **then** set $y^{(t)} = 1$ **else** $y^{(t)} = 0$
  Observe $f^{(t)} \in \{0,1\}$ and record mistakes $m_i^{(t)} = |x_i^{(t)} - f^{(t)}|$ for all $i \in [n]$
  Update $w_i^{(t+1)} = w_i^{(t)} \cdot (1-\eta)^{m_i^{(t)}}$ for all $i \in [n]$.

Majority makes a number of mistakes of at most

$$(2+2\eta)\min_i M_i^T + \frac{2\ln n}{\eta} \ .$$

Proof. We use the sum of weights $W^{(t)}$ to relate our mistakes to the ones of the best expert. First consider our mistakes.

- Consider a round t where we make a mistake.
- Let $U \subseteq [n]$ be the set of classifiers that are wrong in round t.
- By our choice, $\sum_{i\in U} w_i^{(t)} \geq \sum_{i\notin U} w_i^{(t)}$, or equivalently $\sum_{i\in U} w_i^{(t)} \geq W^{(t)}/2$
- All $i \in U$ also made a mistake: $w_i^{(t+1)} = w_i^{(t)}(1-\eta)$ for $i \in U$, and $w_i^{(t+1)} = w_i^{(t)}$ else.
- This implies for the total weight in the next round

$$W^{(t+1)} = \sum_{i\in U}(1-\eta)\cdot w_i^{(t)} + \sum_{i\notin U} w_i^{(t)} = W^{(t)} - \eta\cdot\sum_{i\notin U} w_i^{(t)} \leq \left(1-\frac{\eta}{2}\right)\cdot W^{(t)} \ .$$

- If we make M mistakes in total, then $W^{(T+1)} \leq (1-\eta/2)^M \cdot W^{(1)} = (1-\eta/2)^M n$.

Now consider the number of mistakes of the best expert.

- Let $M_{i^*}^{(T)} = \sum_t m_{i^*}^{(t)}$ be the total number of mistakes of the best expert i^*.
- Then $w_{i^*}^{(t+1)} = w_{i^*}^{(t)} \cdot (1-\eta)$ whenever $m_{i^*}^{(t)} = 1$.
- Hence $w_{i^*}^{(T+1)} = (1-\eta)^{M_{i^*}^{(T)}} \cdot w_{i^*}^{(1)} = (1-\eta)^{M_{i^*}^{(T)}}$
- Relating this to $W^{(T+1)}$ and combining it with the above observation, we see

$$(1-\eta)^{M_{i^*}^{(T)}} \leq W^{(T+1)} \leq \left(1-\frac{\eta}{2}\right)^M n.$$

The inequality relates our mistakes M and the mistakes of the best expert $M_{i^*}^{(T)}$. To extract a more direct upper bound on M, we take a logarithm on both sides

$$M_{i^*}^{(T)}\ln(1-\eta) \leq M\ln\left(1-\frac{\eta}{2}\right) + \ln n.$$

Using the following bounds on the logarithm

$$-z^2 - z \leq \ln(1-z) \leq -z \qquad \text{for all } z \in [0, 0.5], \tag{7.1}$$

Algorithm 17: Randomized Weighted Majority (RWM) for EXPERTS

Set $w_i^1 = 1$ for all $i \in [n]$
for *round* $t = 1, \ldots, T$ **do**
  Set $W^{(t)} = \sum_{i \in [n]} w_i^{(t)}$
  Choose expert i with probability $p_i^{(t)} = w_i^{(t)}/W^t$
  Observe costs $\ell_i^{(t)}$ for all $i \in [n]$
  Update $w_i^{(t+1)} = w_i^{(t)} \cdot (1-\eta)^{\ell_i^{(t)}}$ for all $i \in [n]$.

we obtain

$$M_{i^*}^{(T)}(-\eta^2 - \eta) \leq M(-\eta/2) + \ln n$$

or, equivalently,

$$M \leq (2 + 2\eta) M_{i^*}^{(T)} + (2 \ln n)/\eta \ .$$

□

[Pic: Sandwich-Bounds for $\ln(1-z)$ with $z \in [0, 0.5]$]

Hence, we can expect to make at most ca. twice as many mistakes as the best classifier. Using randomization, we can decrease this factor to 1, i.e., we make a number of mistakes that approaches the one of the best classifier.

Instead of a majority vote, the algorithm now adopts the decision of a single expert. It picks any expert with a probability proportional to the weight $p_i^{(t)} = w_i^{(t)}/W^{(t)}$. As such, successful experts are preferred over bad ones. The RWM algorithm (Algorithm 17) can be used even in a more general setting, the EXPERTS problem:

- There are n experts and T rounds
- Each round t we pick one expert and follow her advice
- *After deciding for an expert*, nature reveals a cost $\ell_i^{(t)} \in [0, 1]$ for each expert.
- We experience the cost of the expert which we chose to follow
- **Goal**: Obtain total cost comparable to the **best expert in hindsight**

In this EXPERTS problem, we recover EXPERTCLASSIFICATION by setting $\ell_i^{(t)} = 1$ whenever the classifier makes a mistake and 0 otherwise.

For the following result, let $L_i^{(T)} = \sum_{t=1}^{T} \ell_i^{(t)}$ denote the total cost of expert i in hindsight, and $L_{RWM}^{(T)}$ denote the expected total cost of RWM.

Theorem 35. *For any sequence of cost vectors from* $[0, 1]$*, the RWM algorithm obtains*

$$L_{RWM}^{(T)} \leq (1 + \eta) \min_i L_i^{(T)} + \frac{\ln n}{\eta} \ .$$

Before we proceed to the proof, let us formally express the strength of this result. We say

$$Regret(T) = L_A^{(T)} - \min_i L_i^{(T)}$$

is the **(external) regret** of algorithm A on a sequence of T cost vectors. Algorithm A is called a **no-(external)-regret** algorithm if $Regret(T) = o(T)$. In such an algorithm, the average regret over time $Regret(T)/T \to 0$ as T grows to infinity. Hence, as time grows, the algorithm achieves the same average cost as the best expert in hindsight.

Observe that setting $\eta = \sqrt{\frac{\ln n}{T}}$ yields

$$L_{RWM}^{(T)} \leq \min_i L_i^{(T)} + 2\sqrt{T \ln n} \ .$$

Corollary 3. *Using $\eta = \sqrt{\frac{\ln n}{T}}$, RWM has external regret at most $2\sqrt{T \ln n}$ and is a no-external-regret algorithm.*

Proof of Theorem 35. We again concentrate on the sum of weights over time. First, consider the total cost of the algorithm.

- Note that
$$W^{(t+1)} \ = \ \sum_{i=1}^{n} w_i^{(t+1)} \ = \ \sum_{i=1}^{n} w_i^{(t)} (1-\eta)^{\ell_i^{(t)}} \ .$$
- Observe that $(1-\eta)^z \ = \ (1 - z\eta)$, for both $z = 0$ and $z = 1$.
- Furthermore, $z \to (1-\eta)^z$ is a convex function for $z \in [0, 1]$, and this implies
$$(1-\eta)^\ell \ \leq \ (1 - \ell\eta).$$
[Pic: Convex]
- This gives
$$W^{(t+1)} \ \leq \ \sum_{i=1}^{n} w_i^{(t)} (1 - \ell_i^{(t)}\eta) = W^{(t)} - \eta \cdot \sum_{i=1}^{n} w_i^{(t)} \ell_i^{(t)}$$
- Let $\ell_{RWM}^{(t)} = \sum_{i=1}^{n} \ell_i^{(t)} w_i^{(t)} / W^{(t)}$, the expected loss of RWM in step t. Substituting this into the bound for $W^{(t+1)}$ gives
$$W^{(t+1)} \ \leq \ W^{(t)} - \eta \ell_{RWM}^{(t)} W^{(t)} \ = \ W^{(t)} (1 - \eta \ell_{RWM}^{(t)}) \ ,$$
and, as a consequence,
$$W^{(T+1)} \ \leq \ W^{(1)} \cdot \prod_{t=1}^{T} (1 - \eta \ell_{RWM}^{(t)}) \ = \ n \cdot \prod_{t=1}^{T} (1 - \eta \ell_{RWM}^{(t)}) \ .$$

As such, $W^{(T+1)}$ is upper bounded in terms of the expected loss of RWM. Let us now turn our attention to the best expert in hindsight.

- For each expert i, the final weight
$$w_i^{(T+1)} = w_i^{(1)} \cdot \prod_{t=1}^{T} (1-\eta)^{\ell_i^{(t)}} = (1-\eta)^{L_i^{(T)}}.$$

- This yields

$$W^{(T+1)} \geq \max_{i\in[n]} w_i^{(T+1)} = (1-\eta)^{\min_i L_i^{(T)}} .$$

As such, $W^{(T+1)}$ is lower bounded in terms of the cost of the best expert. Combining the bounds for $W^{(T+1)}$ and taking the logarithm on both sides gives us

$$\min_i L_i^{(T)} \cdot \ln(1-\eta) \leq (\ln n) + \sum_{t=1}^{T} \ln(1-\eta\ell_{RWM}^{(t)}) .$$

Applying (7.1), we obtain

$$\min_i L_i^{(T)} \cdot (-\eta-\eta^2) \leq (\ln n) + \sum_{t=1}^{T}(-\eta\ell_{RWM}^{(t)}) = (\ln n) - \eta L_{RWM}^{(T)} .$$

Finally, solving for $L_{RWM}^{(T)}$ gives

$$L_{RWM}^{(T)} \leq (1+\eta)\min_i L_i^{(T)} + \frac{\ln n}{\eta} .$$

□

7.2 Multi-Armed Bandits

In the Experts problem, we assume that in every round t, we learn the complete cost vector $\ell^{(t)}$ for all n possible choices (i.e., "experts"). In many applications this is not reasonable – in fact, for most decisions in life we only learn about the outcome after making a choice, and usually we do not learn (entirely) what would have happened if we had decided differently.

The Adversarial Multi-Armed Bandit problem is the Experts problem with such limited feedback:

- n experts, T rounds, costs $\ell_i^{(t)} \in [0,1]$ exactly as in the Experts problem
- Let I_t be the expert chosen by the algorithm in round t.
- I_t is possibly a random variable, depends on choices and observed costs in rounds $t' < t$
- In round t, algorithm only learns $\ell_{I_t}^{(t)}$, but none of $\ell_i^{(t)}$ for $i \neq I_t$.
- **Goal**: Minimize **regret** of the algorithm

Algoithmically, the Adversarial Multi-Armed Bandit problem requires substantially more *exploration*. Algorithm 18 is the Exp3 algorithm (Explore and Exploit with Exponential weights). It uses a variant of the multiplicative weights update from the RWM algorithm with additional exploration and adjusted costs for limited feedback:

- Selection probabilities $q_i^{(t)}$ can be interpreted as follows: With prob. $(1-\gamma)$ we **exploit** based on previous performance and weights, with prob. γ we **explore** uniformly at random.

Algorithm 18: Exp3 Algorithm for Adversarial Multi-Armed Bandit

Set $w_i^1 = 1$ for all $i \in [n]$
for *round* $t = 1, \dots, T$ **do**
  Set $W^{(t)} = \sum_{i \in [n]} w_i^{(t)}$ and $p_i^{(t)} = w_i^{(t)} / W^t$
  Set probabilities $q_i^{(t)} = (1-\gamma) \cdot p_i^{(t)} + \gamma \cdot 1/n$
  Draw I_t according to $q^{(t)}$, observe cost $\ell_{I_t}^{(t)}$
  Set normalized cost $\tilde{\ell}_i^{(t)} = \ell_{I_t}^{(t)} / q_{I_t}^{(t)}$ and $\tilde{\ell}_i^{(t)} = 0$ for all $i \neq I_t$
  Update $w_i^{(t+1)} = w_i^{(t)} \cdot \mathsf{e}^{-\eta \cdot \tilde{\ell}_i^{(t)}}$ for all $i \in [n]$.

- Normalized cost $\tilde{\ell}$ is a "fake-cost" only used for weight update. Designed to address the missing feedback, it can be as large as $\tilde{\ell}_i^{(t)} = \ell_i^{(t)} / q_i^{(t)} \leq 1/(\gamma/n) = n/\gamma$, which might be larger than 1. At first, this might seem excessive. However, $\ell_i^{(t)} \neq 0$ only if expert i is chosen and $\ell_i^{(t)} > 0$. Hence, most of the time, there is no cost for expert i, and hence no update to $w_i^{(t)}$ at all.
- The update uses $\mathsf{e}^{-\eta \cdot \tilde{\ell}_i^{(t)}}$ instead of $(1-\gamma)^{\tilde{\ell}_i^{(t)}}$. Both terms are 1 for $\tilde{\ell}_i^{(t)} = 0$ and monotonically go to 0 as $\tilde{\ell}_i^{(t)}$ goes to infinity. For small γ, they are extremely similar (try plotting them). The first one will be easier for us in the analysis.
- We discuss the exact choices for γ and η later. However, eventually we will have $\eta \leq \gamma/n$, so in all exponents of the update step $\eta \cdot \tilde{\ell}_i^{(t)} \in [0, 1]$.

The main result of this section is the following theorem.

Theorem 36. *If $\eta \leq \gamma/n$, the Exp3-algorithm has an expected cost of at most*

$$L_{Exp3}^{(T)} \leq \min_i L_i^{(T)} + \frac{\ln n}{\eta} + nT \cdot (\eta + \gamma/n) \ .$$

This bound implies the following direct corollary.

Corollary 4. *Using $\eta = \sqrt{\frac{\ln n}{nT}}$ and $\gamma = \eta n$, Exp3 has external regret at most $3\sqrt{nT \ln n}$ and is a no-external-regret algorithm.*

Before proving the theorem, we first establish a technical lemma with a bound on the overall normalized cost multiplied with the $p_i^{(t)}$ stemming from the multiplicative-weights update.

Lemma 23. *Suppose all cost vectors of all rounds $\tilde{\ell}^{(1)}, \dots, \tilde{\ell}^{(T)}$ satisfy $0 \leq \tilde{\ell}_i^{(t)} \leq 1/\eta$ for all $i \in [n]$ and $1 \leq t \leq T$, and let $\tilde{L}_i^{(T)} = \sum_{t=1}^{T} \tilde{\ell}_i^{(t)}$. Then the vectors $p^{(1)}, \dots, p^{(T)}$ computed by the multiplicative-weights update satisfy*

$$\sum_{t=1}^{T} \sum_{i=1}^{n} p_i^{(t)} \tilde{\ell}_i^{(t)} - \eta \sum_{t=1}^{T} \sum_{i=1}^{n} p_i^{(t)} (\tilde{\ell}_i^{(t)})^2 \leq \min_i \tilde{L}_i^{(T)} + \frac{\ln n}{\eta} \ .$$

Proof. The proof is similar to the proof of Theorem 35 above. We again use the weights and relate them to (a) the normalized cost on the left-hand side and (b) every $\tilde{L}_i^{(T)}$. For step (a):

- Consider the weight change $w_i^{(t+1)} = w_i^{(t)} \cdot \mathsf{e}^{-\eta\tilde{\ell}_i^{(t)}}$.
- Note that $\mathsf{e}^z \leq 1 + z + z^2$ for $z \in [-1, 1]$, which implies

$$\begin{aligned}
W^{(t+1)} &= \sum_{i=1}^{n} w_i^{(t+1)} = \sum_{i=1}^{n} w_i^{(t)} \cdot \mathsf{e}^{-\eta\tilde{\ell}_i^{(t)}} \\
&\leq \sum_{i=1}^{n} w_i^{(t)} \cdot (1 - \eta\tilde{\ell}_i^{(t)} + (\eta\tilde{\ell}_i^{(t)})^2) && \text{since } -1 \leq -\eta\tilde{\ell}_i^{(t)} \leq 0 \\
&= \sum_{i=1}^{n} w_i^{(t)} - \sum_{i=1}^{n} w_i^{(t)}\eta\tilde{\ell}_i^{(t)} + \sum_{i=1}^{n} w_i^{(t)}(\eta\tilde{\ell}_i^{(t)})^2) \\
&= W^{(t)} \left(1 - \eta \sum_{i=1}^{n} p_i^{(t)}\tilde{\ell}_i^{(t)} + \eta^2 \sum_{i=1}^{n} p_i^{(t)}(\tilde{\ell}_i^{(t)})^2\right) && \text{since } w_i(t) = W^{(t)} \cdot p_i^{(t)}.
\end{aligned}$$

- Repeatedly applying this equation and using $W^{(1)} = n$ we obtain

$$W^{(T+1)} \leq n \prod_{t=1}^{T} \left(1 - \eta \sum_{i=1}^{n} p_i^{(t)}\tilde{\ell}_i^{(t)} + \eta^2 \sum_{i=1}^{n} p_i^{(t)}(\tilde{\ell}_i^{(t)})^2\right) \quad .$$

Now for step (b), since $w_i^{(t+1)} = w_i^{(t)} \cdot \mathsf{e}^{-\eta\tilde{\ell}_i^{(t)}}$, we obtain recursively

$$W^{(T+1)} \geq \max_i w_i^{(T+1)} = \max_i \left(1 \cdot \prod_{t=1}^{T} \mathsf{e}^{-\eta\tilde{\ell}_i^{(t)}}\right) = \max_i \mathsf{e}^{-\eta\sum_{t=1}^{T}\tilde{\ell}_i^{(t)}} = \mathsf{e}^{-\eta\min_i \tilde{L}_i^{(T)}}.$$

Applying the bounds on $W^{(T+1)}$ we see

$$\mathsf{e}^{-\eta\min_i \tilde{L}_i^{(T)}} \leq W^{(T+1)} \leq n \prod_{t=1}^{T} \left(1 - \eta \sum_{i=1}^{n} p_i^{(t)}\tilde{\ell}_i^{(t)} + \eta^2 \sum_{i=1}^{n} p_i^{(t)}(\tilde{\ell}_i^{(t)})^2\right)$$

and apply a logarithm on both sides

$$-\eta \min_i \tilde{L}_i^{(T)} \leq \ln n + \sum_{t=1}^{T} \ln \left(1 - \eta \sum_{i=1}^{n} p_i^{(t)}\tilde{\ell}_i^{(t)} + \eta^2 \sum_{i=1}^{n} p_i^{(t)}(\tilde{\ell}_i^{(t)})^2\right).$$

Using $\ln(1 + z) \leq z$ for all $z > -1$, we can further increase the right-hand side to obtain

$$-\eta \min_i \tilde{L}_i^{(T)} \leq \ln n + \sum_{t=1}^{T} \left(-\eta \sum_{i=1}^{n} p_i^{(t)}\tilde{\ell}_i^{(t)} + \eta^2 \sum_{i=1}^{n} p_i^{(t)}(\tilde{\ell}_i^{(t)})^2\right).$$

Dividing both sides by η and rearraging the terms yields the statement of the lemma. □

Using the lemma, we can now start to prove the main theorem.

Proof of Theorem 36. Let us first fix the behavior of the algorithm and consider what happens for each set of possible costs and coin flips by the algorithm

- We fix the cost vectors $\ell^{(1)}, \dots, \ell^{(T)}$.
- Let us also fix the random choices of experts by the algrithm $I_1, \dots, I_T$.
- This fixes all normalized cost vectors $\tilde{\ell}^{(1)}, \dots, \tilde{\ell}^{(T)}$.
- Ultimately, this also fixes all vectors $p^{(1)}, \dots, p^{(T)}$ and $q^{(1)}, \dots, q^{(T)}$.
- Having fixed all these values, we see

$$\begin{aligned}
\sum_{t=1}^{T}\sum_{i=1}^{n} q_i(t)\tilde{\ell}_i^{(t)} &= \sum_{t=1}^{T}\sum_{i=1}^{n}\left((1-\gamma)p_i^{(t)} + \frac{\gamma}{n}\right)\cdot \tilde{\ell}_i^{(t)} \\
&= (1-\gamma)\sum_{t=1}^{T}\sum_{i=1}^{n} p_i^{(t)}\tilde{\ell}_i^{(t)} + \frac{\gamma}{n}\sum_{t=1}^{T}\sum_{i=1}^{n}\tilde{\ell}_i^{(t)} \\
&\le (1-\gamma)\left(\min_i \tilde{L}_i^{(T)} + \frac{\ln n}{\eta} + \eta\sum_{t=1}^{T}\sum_{i=1}^{n} p_i^{(t)}(\tilde{\ell}_i^{(t)})^2\right) + \frac{\gamma}{n}\sum_{t=1}^{T}\sum_{i=1}^{n}\tilde{\ell}_i^{(t)} \\
&\le \min_i \tilde{L}_i^{(T)} + \frac{\ln n}{\eta} + \eta\sum_{t=1}^{T}\sum_{i=1}^{n} q_i^{(t)}(\tilde{\ell}_i^{(t)})^2 + \frac{\gamma}{n}\sum_{t=1}^{T}\sum_{i=1}^{n}\tilde{\ell}_i^{(t)},
\end{aligned} \tag{7.2}$$

where in the first inequality follows from the lemma above.

Now let us bring back some of the randomness to relate the costs $\ell_i^{(t)}$ and $\tilde{\ell}_i^{(t)}$. Keeping all costs $\ell^{(t)}$ fixed, we only fix the choices $I_1, \dots, I_{t-1}$ before round t. In round t, for every $i \in [n]$

$$\begin{aligned}
\mathbb{E}\left[\tilde{\ell}_i^{(t)} \mid I_1, \dots, I_{t-1}\right] &= \Pr[I_t = i \mid I_1, \dots, I_{t-1}] \cdot \frac{\ell_i^{(t)}}{q_i^{(t)}} + \Pr[I_t \neq i \mid I_1, \dots, I_{t-1}] \cdot 0 \\
&= q_i^{(t)} \cdot \frac{\ell_i^{(t)}}{q_i^{(t)}} + 0 \;=\; \ell_i^{(t)}.
\end{aligned}$$

As this is independent of $I_1, \dots, I_{t-1}$, we can directly bound the unconditional expectation

$$\mathbb{E}\left[\sum_{i\in[n]} q_i^{(t)}\tilde{\ell}_i^{(t)}\right] = \sum_{i\in[n]}\mathbb{E}\left[q_i^{(t)}\tilde{\ell}_i^{(t)}\right] = \sum_{i\in[n]}\mathbb{E}\left[q_i^{(t)}\right]\ell_i^{(t)} = \mathbb{E}\left[\ell_{I_t}^{(t)}\right] = \ell_{Exp3}^{(t)}.$$

As such, the expectation of the left-hand side of (7.2) is exactly the expected cost $L_{Exp3}^{(T)}$ of the algorithm.

Now for the last right-hand side of (7.2) we have to bound the quadratic terms. Fixing the choices $I_1, \dots, I_{t-1}$ before round t arbitrarily, we see

$$\mathbb{E}\left[(\tilde{\ell}_i^{(t)})^2 \mid I_1, \dots, I_{t-1}\right] \;=\; q_i^{(t)} \cdot \left(\frac{\ell_i^{(t)}}{q_i^{(t)}}\right)^2 + 0 = \frac{(\ell_i^{(t)})^2}{q_i^{(t)}} \;.$$

Since the expression has $q_i^{(t)}$ in the denominator, it is not independent of $I_1, \dots, I_{t-1}$. However, we see that

$$\mathbb{E}\left[\sum_{i\in[n]} q_i^{(t)}(\tilde{\ell}_i^{(t)})^2 \;\middle|\; I_1, \dots, I_{t-1}\right] = \sum_{i\in[n]} q_i^{(t)} \cdot \frac{(\ell_i^{(t)})^2}{q_i^{(t)}} = \sum_{i\in[n]}(\ell_i^{(t)})^2$$

is independent of $I_1, \ldots, I_{t-1}$ and, thus, also applies to the unconditional expectation

$$\mathbb{E}\left[\sum_{i\in[n]} q_i^{(t)}(\tilde{\ell}_i^{(t)})^2\right] = \sum_{i\in[n]}(\ell_i^{(t)})^2 \ .$$

Overall, using random $I_1, \ldots, I_T$, the expectation over equation (7.2) is

$$\sum_{t=1}^{T}\sum_{i=1}^{n}\mathbb{E}\left[q_i(t)\tilde{\ell}_i^{(t)}\right] \ \leq \ \mathbb{E}\left[\min_i \tilde{L}_i^{(T)}\right] + \frac{\ln n}{\eta} + \eta\sum_{t=1}^{T}\sum_{i=1}^{n}\mathbb{E}\left[q_i^{(t)}(\tilde{\ell}_i^{(t)})^2\right] + \frac{\gamma}{n}\sum_{t=1}^{T}\sum_{i=1}^{n}\mathbb{E}[\tilde{\ell}_i^{(t)}].$$

Replacing the terms using the observations above yields

$$L_{Exp3}^{(T)} \ \leq \ \min_i L_i^{(T)} + \frac{\ln n}{\eta} + \eta\sum_{t=1}^{T}\sum_{i=1}^{n}(\ell_i^{(t)})^2 + \frac{\gamma}{n}\sum_{t=1}^{T}\sum_{i=1}^{n}\ell_i^{(t)}.$$

Finally, since $\ell_i^{(t)} \in [0,1]$, the two double sums are upper bounded by nT. This is not overly costly when η and γ/n are sufficiently small (but not too small, since then $(\ln n)/\eta$ becomes large). Hence,

$$L_{Exp3}^{(T)} \ \leq \ \min_i L_i^{(T)} + \frac{\ln n}{\eta} + nT\cdot(\eta + \gamma/n),$$

which proves the theorem. □

7.3 Online Convex Optimization

In the previous section, we considered problems with a finite set of experts. In many problems, however, the set of possible alternatives is infinite, in which case the algorithms above cannot be directly applied. In this section, we will consider problems in which experts are numerical vectors from a convex set, and costs are given by convex functions.

Online Convex Optimization is an experts problem with infinitely many experts:

- Experts are vectors in a convex and compact set $D \subset \mathbb{R}^d$
- Every round $t = 1, \ldots, T$ we pick a point $\mathbf{w}^{(t)} \in D$
- Then we see a convex cost function $c_t : D \to \mathbb{R}$ and incur cost $c_t(\mathbf{w}^{(t)})$.
- **Goal:** Pick $\mathbf{w}^{(t)}$ to minimize total cost $\sum_{t=1}^{T} c_t(\mathbf{w}(t))$.
- We again evaluate our performance using a notion of regret

$$Regret(T) = \sum_{t=1}^{T} c_t(\mathbf{w}^{(t)}) - \min_{\mathbf{v}\in D}\sum_{t=1}^{T} c_t(\mathbf{v}) \ .$$

Example 11. Consider a set of data points $(x_1, y_1), \ldots, (x_T, y_T)$ in two-dimensional space. We look for a linear regression, i.e., a line $w_1x + w_2$ that minimizes the sum of squared error $\sum_t (w_1x_t + w_2 - y_t)^2$. The set of possible choices ("experts") is given by all pairs $(w_1, w_2) \in \mathbb{R}^2$.

Consider an online variant where points arrive one by one. Upon seeing x_t, we have to predict y_t by choosing a linear function with parameters $(w_1^{(t)}, w_2^{(t)})$. Then y_t is revealed, which implies the squared-error cost

$$c_t(w_1^{(t)}, w_2^{(t)}) = (w_1^{(t)} x_t + w_2^{(t)} - y_t)^2 \ .$$

The cost is convex and differentiable in the arguments. The best line in hindsight is just the optimal regression. ■

7.3.1 Generalized Infinitesimal Gradient Ascent

In this section, we will assume that the functions c_t are **differentiable** and we also **learn the gradient**. This is not really necessary, but it simplifies the analysis a lot. The *gradient* at a point $\mathbf{w} \in \mathbb{R}^d$ is the vector $\nabla c(\mathbf{w}) \in \mathbb{R}^d$ of partial derivatives $\frac{\partial c}{\partial x_i}$. For example, for the regression above,

$$\nabla c_t(w_1, w_2) = \begin{pmatrix} 2x_t^2 w_1 + 2x_t w_2 - 2x_t y_t \\ 2x_t w_1 + w_2 - y_t \end{pmatrix} \ . \tag{7.3}$$

To gain some intuition for the algorithm, consider minimizing a *single* convex function c.

- We can use gradient descent – start at any point $\mathbf{w}^{(1)} \in \mathbb{R}^d$ and make steps in the direction of steepest descent

  $$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla c(\mathbf{w}^{(t)}) \ ,$$

  with sufficiently small step size $\eta \in \mathbb{R}$. We "sink into the valley" of the convex function and arrive close to (depending on η) the global minimum.
- Gradient descent to minimize c **over a convex subspace** D can lead us outside D. When this happens, we use a **projection** $P : \mathbb{R}^d \to D$ that finds for any $\mathbf{w} \in \mathbb{R}^d$ the point $P(\mathbf{w}) \in D$ closest to $\mathbf{w}$.
- Using projected gradient updates

  $$\mathbf{w}^{(t+1)} = P(\mathbf{w}^{(t)} - \eta \nabla c(\mathbf{w}^{(t)})) \ ,$$

  the convexity of D ensures that we "circle around the border" of D to sink towards the minimum of c within D.

For the online scenario with changing functions, we use the **generalized infinitesimal gradient ascent**[1] **(GIGA)** algorithm:

$$\text{Pick } \mathbf{w}^{(1)} \in D \text{ arbitrary} \quad \text{and} \quad \mathbf{w}^{(t+1)} = P(\mathbf{w}^{(t)} - \eta \cdot \nabla c_t(\mathbf{w}^{(t)})) \ .$$

It uses the gradient for c_t to optimize for c_{t+1}. This seems like a stupid idea, as c^{t+1} can be completely different from c_t. Nevertheless, it turns out that this algorithm has small regret, at least if diameter of D and steepness of functions c_t are bounded independent of T.

[1] Ascent stems from the equivalent formulation of the scenario as online *concave maximiziation*.

Theorem 37. *Let $G \geq \|\nabla c_t(\mathbf{w})\|_2$ and $\Delta \geq \|\mathbf{v} - \mathbf{w}\|_2$ for all $\mathbf{v}, \mathbf{w} \in D$. GIGA experiences a regret of at most*

$$\frac{\Delta^2}{2\eta} + \frac{T\eta G^2}{2} \ .$$

Corollary 5. *Using $\eta = \Delta/(G\sqrt{T})$, the regret of GIGA is bounded by $\Delta G\sqrt{T}$. If Δ and G are independent of T, GIGA is a no-regret algorithm.*

Intuitively, projected gradient descent works if all cost functions c_t are similar. If functions are highly different, the algorithm can get high cost, but then the optimum $\mathbf{w}^* \in D$ must have high cost, too. We use a potential function argument to capture this intuition:

- W.l.o.g. label optimum as origin $\mathbf{w}^* = 0$ of the coordinate system.
- Consider "potential" as $\Phi_t = \frac{1}{2\eta}\|\mathbf{w}^{(t)}\|_2^2$.
- Φ_t measures distance of $\mathbf{w}^{(t)}$ to $\mathbf{w}^* = 0$

Our first lemma expresses a trade-off. Either the cost in a round is close to optimal, or the next vector $\mathbf{w}^{(t+1)}$ is closer to $\mathbf{w}^*$ than $\mathbf{w}^{(t)}$.

Lemma 24. $c_t(\mathbf{w}(t)) - c_t(0) + \Phi_{t+1} - \Phi_t \leq \eta \cdot G^2/2$

Proof. Note that $\|P(\mathbf{v})\|_2 \leq \|\mathbf{v}\|_2$, because D is convex and the projection always moves $\mathbf{v}$ towards D and, thus, the origin $\mathbf{w}^* = 0 \in D$. Now we see

$$\begin{aligned}
&\Phi_{t+1} - \Phi_t \\
&= \frac{1}{2\eta}(\|\mathbf{w}^{(t+1)}\|_2^2 - \|\mathbf{w}^{(t)}\|_2^2) \\
&\leq \frac{1}{2\eta}(\|\mathbf{w}^{(t)} - \eta\nabla c_t(\mathbf{w}^{(t)})\|_2^2 - \|\mathbf{w}^{(t)}\|_2^2) && \text{since } \|P(\mathbf{w})\|_2 \leq \|\mathbf{w}\|_2 \\
&= \frac{1}{2\eta}(\|\mathbf{w}^{(t)}\|_2^2 + \eta^2\|\nabla c_t(\mathbf{w}^{(t)})\|_2^2 - 2\langle\eta\nabla c_t(\mathbf{w}^{(t)}), \mathbf{w}^{(t)}\rangle - \|\mathbf{w}^{(t)}\|_2^2) && \text{vector law of cosines} \\
&\leq \frac{1}{2}\eta G^2 - \langle\nabla c^t(\mathbf{w}^{(t)}), \mathbf{w}^{(t)}\rangle \ . && \text{by definition of } G
\end{aligned}$$

The vector law of cosines is a vector version of the first binomial formula

$$\|\mathbf{v} + \mathbf{w}\|_2^2 = \|\mathbf{v}\|_2^2 + \|\mathbf{w}\|_2^2 + 2\langle\mathbf{v}, \mathbf{w}\rangle$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product given by $\langle\mathbf{x}, \mathbf{y}\rangle = \sum_{i=1}^d x_i y_i$.

Convexity means a function always stays above its' tangents. For a one-dimensional convex function, we can easily see

$$f(v) \geq f(w) + f'(w) \cdot (v - w),$$

and the analogous observation holds for our convex cost functions in multiple dimensions

$$c_t(\mathbf{v}) \geq c_t(\mathbf{w}^{(t)}) + \langle\nabla c_t(\mathbf{w}^{(t)}), (\mathbf{v} - \mathbf{w}^{(t)})\rangle \ .$$

Using $\mathbf{v} = \mathbf{w}^* = 0$ and rearranging terms

$$\langle\nabla c_t(\mathbf{w}^{(t)}), (0 - \mathbf{w}^{(t)})\rangle \leq c_t(0) - c_t(\mathbf{w}^{(t)})$$

and we see

$$\Phi_{t+1} - \Phi_t \leq \frac{1}{2}\eta G^2 - \langle \nabla c^t(\mathbf{w}^{(t)}), \mathbf{w}^{(t)} \rangle \leq \eta G^2/2 + c_t(0) - c_t(\mathbf{w}^{(t)}) \ ,$$

which proves the lemma. □

Proof of Theorem 37. Summing up from $t = 1, \ldots, T$, we get a telescopic sum and the lemma yields

$$\sum_{t=1}^{T} (c_t(\mathbf{w}^{(t)}) - c_t(0) + \Phi_{t+1} - \Phi_t) \ = \ \Phi_{T+1} - \Phi_1 + \sum_{t=1}^{T} (c_t(\mathbf{w}^{(t)}) - c_t(0)) \ \leq \ T \cdot \eta G^2/2 \ .$$

Note that, by definition, $\frac{\Delta^2}{2\eta} \geq \Phi_t \geq 0$, so the regret

$$\begin{aligned} \sum_{t=1}^{T} c_t(\mathbf{w}^{(t)}) - \min_{\mathbf{v} \in D} \sum_{t=1}^{T} c_t(\mathbf{v})) &= \sum_{t=1}^{T} c_t(\mathbf{w}^{(t)}) - c_t(0) \leq \Phi_1 - \Phi_{T+1} + \frac{T\eta G^2}{2} \\ &\leq \frac{\Delta^2}{2\eta} + \frac{T\eta G^2}{2} \end{aligned}$$

□

7.3.2 Follow-the-Regularized-Leader

A related approach to the problem is to *Follow The Leader (FTL)*, i.e., in each round we pick $\mathbf{w}^{(t)} \in D$ as the vector that has experienced smallest total cost in rounds $1, \ldots, t-1$. Unfortunately, this strategy can be tricked very easily in the adversarial setting, by simply giving $\mathbf{w}^{(t)}$ a very high cost in the next round. FTL can become very unstable and alternating in its' choices. It can guarantee only a high cost, since it might be "always too late".

Intuitively, one would want a more gentle adaptation, which nevertheless manages to identify good choices rapidly over time (whenever they exist). For this we add a regularization term defined by a function $R : D \to \mathbb{R}$ – a bit of "noise", or a hypothetical round 0 with additional cost. It is added when we compute the best choice so far. Carefully chosen regularization terms can indeed stabilize FTL and eventually turn it into a no-regret algorithm.

Follow-The-Regularized-Leader (FTRL) picks in each round t a vector $\mathbf{w}^{(t)}$ that minimizes the cost $R(\mathbf{w}) + \sum_{t'=1}^{t-1} c_{t'}(\mathbf{w})$. For simplicity of exposition we use $c_0(\mathbf{w}) = R(\mathbf{w})$. The next lemma bounds the regret in terms of $c_t(\mathbf{w}^{(t)}) - c_t(\mathbf{w}^{t+1})$. The regret is governed by the points chosen in rounds t and $t+1$ and their cost difference measured in round t, for all $t \geq 1$.

Lemma 25. *The regret of FTRL is upper bounded by*

$$\max_{\mathbf{v} \in D} R(\mathbf{v}) - R(\mathbf{w}^{(1)}) + \sum_{t=1}^{T} (c_t(\mathbf{w}^{(t)}) - c_t(\mathbf{w}^{(t+1)})) \ .$$

Proof. We use the interpretation that $R(\mathbf{w}) = c_0(\mathbf{w})$. First, we show by induction on T that

$$\min_{\mathbf{v}\in D} \sum_{t=0}^{T} c_t(\mathbf{v}) \geq \sum_{t=0}^{T} c_t(\mathbf{w}^{(t+1)}).$$

This is trivial for $T = -1$. Suppose the statement holds for $T-1$. Then

$$\sum_{t=0}^{T-1} c_t(\mathbf{w}^{(T+1)}) \geq \min_{\mathbf{v}\in D} \sum_{t=0}^{T-1} c_t(\mathbf{v}) \geq \sum_{t=0}^{T-1} c_t(\mathbf{w}^{(t+1)})$$

where the second inequality is our hypothesis. Now add $c_T(\mathbf{w}^{(T+1)})$ on both sides,

$$\sum_{t=0}^{T} c_t(\mathbf{w}^{(T+1)}) \geq \sum_{t=0}^{T} c_t(\mathbf{w}^{(t+1)})$$

and note that FTRL picks $w^{(T+1)}$ as the vector that minimizes the total cost $\sum_{t=0}^{T} c_t(\mathbf{w}^{(T+1)})$ of all previous rounds. Hence,

$$\sum_{t=0}^{T} c_t(\mathbf{w}^{(T+1)}) = \min_{\mathbf{v}\in D} \sum_{t=0}^{T} c_t(\mathbf{v}) \geq \sum_{t=0}^{T} c_t(\mathbf{w}^{(t+1)})$$

is proved.

Now using the above inequality, we see that

$$\sum_{t=0}^{T} c_t(\mathbf{w}^t) - \min_{\mathbf{v}\in D} \sum_{t=0}^{T} c_t(\mathbf{v}) \leq \sum_{t=0}^{T} c_t(\mathbf{w}^t) - \sum_{t=0}^{T} c_t(\mathbf{w}^{(t+1)}).$$

Due to the minimum, the bound holds also for every fixed vector $\mathbf{v} \in D$. Replacing c_0 by R,

$$R(\mathbf{w}^{(0)}) - R(\mathbf{v}) + \sum_{t=1}^{T} c_t(\mathbf{w}^t) - \sum_{t=1}^{T} c_t(\mathbf{v}) \leq R(\mathbf{w}^{(0)}) - R(\mathbf{w}^{(1)}) + \sum_{t=1}^{T} c_t(\mathbf{w}^t) - \sum_{t=1}^{T} c_t(\mathbf{w}^{(t+1)}).$$

for any $\mathbf{v} \in D$. Again, noting that the bound holds for all $\mathbf{v} \in D$, rearranging gives the desired upper bound on the regret

$$\begin{aligned}\sum_{t=1}^{T} c_t(\mathbf{w}^{(t)}) - \min_{\mathbf{v}\in D} \sum_{t=1}^{T} c_t(\mathbf{v}) &= \max_{\mathbf{v}\in D} \sum_{t=1}^{T} (c_t(\mathbf{w}^{(t)}) - c_t(\mathbf{v})) \\ &\leq \max_{\mathbf{v}\in D} R(\mathbf{v}) - R(\mathbf{w}^{(1)}) + \sum_{t=1}^{T} (c_t(\mathbf{w}^{(t)}) - c_t(\mathbf{w}^{t+1})).\end{aligned}$$

□

The bound depends on the change of the vectors $\mathbf{w}^{(t)}$ and $\mathbf{w}^{(t+1)}$, **both measured in terms of cost** c_t. The behavior of FTRL is governed by the best choices so far. If that choice changes frequently, then $c_t(\mathbf{w}^t) - c_t(\mathbf{w}^{t+1})$ can become large, and FTRL tends to suffer from a large regret. Here the terms $R(\mathbf{v})$ come to the rescue – they ensure that FTRL changes only when the improvement in terms of cost is substantial. While this increases the regret by at most $\max_{\mathbf{v}\in D} R(\mathbf{v})$, it also leads to more stable behavior and less regret over the rounds.

Example 12. Consider the convex space $D = \{\mathbf{w} \mid \sum_{i=1}^{d} w_i = 1, w_i \in [0,1] \text{ for all } i \in [d]\}$. Suppose in each round, we get a linear cost function based on a vector $\ell^{(t)} \in [0,1]^d$

$$c_t(\mathbf{w}) = \sum_{i=1}^{d} w_i \cdot \ell_i^{(t)} \ .$$

This is **exactly the Experts problem**! The vector $\mathbf{w}$ is a probability distribution over experts, and $c_t(\mathbf{w})$ is the expected cost. As such, online convex optimization is a generalization of the Experts problem. ■

Example 13. Popular choices for the regularizer $R(\mathbf{v})$ are the *Euclidean regularizer*

$$R(\mathbf{v}) = \frac{1}{2\eta} \sum_{i=1}^{d} v_i^2$$

or the *Entropical regularizer*

$$R(\mathbf{v}) = \frac{1}{\eta} \sum_{i=1}^{d} v_i \ln v_i.$$

$\eta > 0$ is a scaling factor – smaller $\eta \Rightarrow$ larger impact of regularization in the algorithm.

In the Experts problem (c.f. Example 12), it can be shown that Entropical regularization results in $\mathbf{w}^{(t)}$ being proportional to $\mathsf{e}^{-\eta L_i^{(t-1)}}$. FTRL becomes a version of the RWM algorithm. ■

To obtain a non-trivial upper bound on the regret of FTRL, we make two further assumptions. The first assumption is on the cost functions c_t.

- We consider a **norm** $\|\cdot\|$ that assigns each point in D a "length".
- Typical examples are the ℓ_p-norms, for $p \in \{1, 2, \ldots\}$

$$\|\mathbf{w}\|_p = \sqrt[p]{\sum_{i=1}^{d} |w_i|^p}.$$

- Based on a norm, we assume all costs c_t satisfy a **Lipschitz condition**. Formally, for all vectors $\mathbf{v}, \mathbf{w} \in D$

$$c_t(\mathbf{v}) - c_t(\mathbf{w}) \ \leq \ L \cdot \|\mathbf{v} - \mathbf{w}\| \ .$$

The second assumption is on the regularization function.

- Assume $R(\mathbf{v})$ is **differentiable** (mostly for simplicity of exposition) and σ**-strongly convex**:
$$R(\mathbf{v}) \;\geq\; R(\mathbf{w}) + \langle \nabla R(\mathbf{w}), \mathbf{v} - \mathbf{w} \rangle + \frac{\sigma}{2}\|\mathbf{v} - \mathbf{w}\|^2 \;\; .$$
- For a strongly convex function, every tangent hyperplane meets the function in exactly one point. Intuitively, the function is always "curved" and "superlinear".
- If $c_1, \ldots, c_T$ are convex and R is *strongly* convex, then $R + \sum_{t'<t} c_{t'}$ is *strongly* convex, for every $t' \in \{1, \ldots, T\}$.
- Hence, if R is strongly convex, then FTRL chooses the "regularized leader" in every round by optimizing a strongly convex function.

Example 12 (continued)**.** Recall the EXPERTS problem. For the ℓ_1 norm

$$c_t(\mathbf{v}) - c_t(\mathbf{w}) = \sum_{i\in[n]} \ell_i^{(t)} \cdot (v_i - w_i) \leq \sum_{i\in[n]} 1 \cdot |v_i - w_i| = 1 \cdot \|\mathbf{v} - \mathbf{w}\|_1$$

and the Lipschitz condition is satisfied with $L = 1$. More generally, if all costs are from $\ell_i^{(t)} \in [0, \rho]$, then $L = \rho$ is sufficient.

For the ℓ_2 norm, $L = \sqrt{n}$ is sufficient:

$$c_t(\mathbf{v}) - c_t(\mathbf{w}) \leq \sum_{i\in[n]} |v_i - w_i| \leq \sqrt{n} \cdot \sqrt{\sum_{i\in[n]} |v_i - w_i|^2} = \sqrt{n}\|\mathbf{v} - \mathbf{w}\|_2 \;\; .$$

■

Example 14. The Euclidean regularizer is $\frac{1}{\eta}$-strongly convex with the ℓ_2 norm. To see this, note that the partial derivative is $(\nabla R(\mathbf{w}))_i = \frac{\partial R}{\partial w_i} = w_i/\eta$ and, hence,

$$\langle \nabla R(\mathbf{w}), \mathbf{v} - \mathbf{w} \rangle = \sum_{i=1}^{d} \frac{w_i}{\eta} \cdot (v_i - w_i) = \frac{1}{\eta} \sum_{i=1}^{d} w_i(v_i - w_i).$$

Thus, overall

$$\begin{aligned}
\langle \nabla R(\mathbf{w}), \mathbf{v} - \mathbf{w} \rangle + \frac{1}{2\eta}\|\mathbf{v} - \mathbf{w}\|_2^2 &= \frac{1}{\eta} \sum_{i=1}^{d} w_i(v_i - w_i) + \frac{1}{2\eta} \sum_{i=1}^{d} (v_i - w_i)^2 \\
&= \frac{1}{2\eta} \sum_{i=1}^{d} v_i^2 - \frac{1}{2\eta} \sum_{i=1}^{d} w_i^2 \\
&= R(\mathbf{v}) - R(\mathbf{w}).
\end{aligned}$$

■

Suppose the Lipschitz condition and σ-strong convexity of the regularizer hold with respect to the same norm. For this case, we obtain the main result of this section.

Theorem 38. *If the regularizer R is σ-strongly convex and each c_t fulfills the Lipschitz condition with parameter L, the regret of FTRL is bounded by*

$$\max_{\mathbf{v}\in D} R(\mathbf{v}) - R(\mathbf{w}^{(1)}) + T \cdot \frac{L^2}{\sigma} \;\; .$$

To prove the theorem, we first show a property of strongly convex functions.

Lemma 26. *Let $F : D \to \mathbb{R}$ be a σ-strongly convex differentiable function over D with respect to a norm $\|\cdot\|$. Let $\mathbf{w} \in \arg\min_{\mathbf{v}\in D} F(\mathbf{v})$. Then, for all $\mathbf{v} \in D$*

$$F(\mathbf{v}) - F(\mathbf{w}) \geq \frac{\sigma}{2}\|\mathbf{v} - \mathbf{w}\|^2 \ .$$

Proof. Suppose $\mathbf{w}$ is in the interior of D.

- Gradient of $F(\mathbf{w})$ must be zero in every component – it is a local minimum, $\nabla F(\mathbf{w}) = 0$.
- By strong convexity, for all $\mathbf{v} \in D$

$$F(\mathbf{v}) - F(\mathbf{w}) \geq \langle \nabla F(\mathbf{w}), \mathbf{v} - \mathbf{w}\rangle + \frac{\sigma}{2}\|\mathbf{v} - \mathbf{w}\|^2 = \frac{\sigma}{2}\|\mathbf{v} - \mathbf{w}\|^2 \ .$$

Suppose $\mathbf{w}$ is not in the interior of D. Then $\langle \nabla F(\mathbf{w}), \mathbf{v} - \mathbf{w}\rangle \geq 0$, since otherwise moving from $\mathbf{w}$ slightly to $\mathbf{v}$ would decrease F. □

Proof of Theorem 38. We prove the theorem by showing that $c_t(\mathbf{w}^{(t)}) - c_t(\mathbf{w}^{(t+1)}) \leq L^2/\sigma$ for all $t \in [1, T]$. The theorem then follows using Lemma 25.

- For all $t \geq 1$, let $F_t(\mathbf{v}) = R(v) + \sum_{t'=1}^{t-1} c_{t'}(\mathbf{v})$. R strongly convex $\Rightarrow$ F_t strongly convex.
- $\mathbf{w}^{(t)}$ is a vector that minimizes F_t. Therefore, by Lemma 26

$$\frac{\sigma}{2}\|\mathbf{w}^{(t+1)} - \mathbf{w}^{(t)}\|^2 \leq F_t(\mathbf{w}^{(t+1)}) - F_t(\mathbf{w}^{(t)}) \ .$$

- Apply the same bound to F_{t+1}, which is minimized at $\mathbf{w}^{(t+1)}$

$$\frac{\sigma}{2}\|\mathbf{w}^{(t)} - \mathbf{w}^{(t+1)}\|^2 \leq F_{t+1}(\mathbf{w}^{(t)}) - F_{t+1}(\mathbf{w}^{(t+1)}) \ .$$

- We sum up these inequalities

$$\sigma \cdot \|\mathbf{w}^{(t)} - \mathbf{w}^{(t+1)}\|^2 \leq (F_t(\mathbf{w}^{(t+1)}) - F_t(\mathbf{w}^{(t)})) + (F_{t+1}(\mathbf{w}^{(t)}) - F_{t+1}(\mathbf{w}^{(t+1)})) \ ,$$

  apply definition of F_t and F_{t+1}, and consider the Lipschitz condition to obtain

$$\sigma \cdot \|\mathbf{w}^{(t)} - \mathbf{w}^{(t+1)}\|^2 \ \leq \ c_t(\mathbf{w}^{(t)}) - c_t(\mathbf{w}^{(t+1)}) \ \leq \ L \cdot \|\mathbf{w}^{(t)} - \mathbf{w}^{(t+1)}\| \ .$$

- This implies

$$\|\mathbf{w}^{(t)} - \mathbf{w}^{(t+1)}\| \leq \frac{L}{\sigma}$$

  and, thus,

$$c_t(\mathbf{w}^{(t)}) - c_t(\mathbf{w}^{(t+1)}) \leq \frac{L^2}{\sigma} \ .$$

Applying this bound in Lemma 25 proves the theorem. □

Example 15. Recall that FTRL with the Entropical regularizer $R(\mathbf{v}) = \frac{1}{\eta}\sum_{i=1}^{n} v_i \ln v_i$ in the Experts problem yields a variant of the RWM algorithm (c.f. Example 13).

- One can show that the Entropical regularizer is $1/\eta$-strongly convex w.r.t. the ℓ_1-norm.
- Also, $-(\ln n)/\eta \leq R(\mathbf{w}) \leq 0$.

- Observed above: Lipschitz condition holds for the ℓ_1 norm and $L = 1$.
- Hence, Theorem 38 yields a regret at most $(\ln n)/\eta + T\eta$.
- Setting $\eta = \sqrt{\frac{\ln n}{T}}$, the bound becomes $2\sqrt{T \ln n}$, exactly the one shown in Theorem 35.

■

Example 16. Consider FTRL with Euclidean regularizer $R(\mathbf{v}) = \frac{1}{2\eta}\sum_{i=1}^{n} v_i^2$ in the EXPERTS problem.

- Recall that the Euclidean regularizer is $\frac{1}{\eta}$-strongly convex for the ℓ_2 norm, i.e., $\sigma = 1/\eta$.
- Also, $0 < R(\mathbf{v}) \leq \frac{1}{2\eta}$.
- Observed above: the Lipschitz condition holds for the ℓ_2-norm and $L = \sqrt{n}$.
- Hence, Theorem 38 yields a regret at most $1/(2\eta) + Tn\eta$.
- Setting $\eta = 1/\sqrt{2nT}$, the bound becomes $\sqrt{2nT}$.

FTRL with Euclidean regularizer is also a no-regret algorithm for the EXPERTS problem. ■

A disadvantage might be that FTRL requires to solve a potentially complicated problem: Find the best vector $\mathbf{w}$ for the sum of costs of all previous rounds. However, in many cases this choice is much simpler than it sounds.

Example 16 (continued and generalized)**.** We consider $D = \mathbb{R}^d$ with linear functions

$$c_t(\mathbf{w}) = \sum_{i=1}^{d} \ell_i^{(t)} w_i \ .$$

This scenario contains the EXPERTS problem as (very) special case. Consider FTRL with Euclidean regularizer.

- We need to find $\mathbf{w}^{(t+1)}$ so as to minimize

$$\sum_{t'=1}^{t}\sum_{i=1}^{d} \ell_i^{(t')} w_i + \frac{1}{2\eta}\sum_{i=1}^{d} w_i^2 \ .$$

- The partial derivative by w_i is

$$\sum_{t'=1}^{t} \ell_i^{(t')} + \frac{1}{\eta} w_i \ .$$

- For a minimum, $w_i^{(t+1)}$ has bring this partial derivative to zero: $w_i^{(t+1)} = -\eta \sum_{t'=1}^{t} \ell_i^{(t')}$. Overall $\mathbf{w}^{(t+1)} = -\eta \sum_{t'=1}^{t} \ell^{(t')}$.
- Applying this property recursively, we see

$$\mathbf{w}^{(t+1)} \ = \ \mathbf{w}^{(t)} - \eta \ell^{(t)} \ = \ \mathbf{w}^{(t)} - \eta \nabla c_t(\mathbf{w}^{(t)}) \qquad \text{for every } \mathbf{w}^{(t)}.$$

- Hence, we obtain $\mathbf{w}^{(t+1)}$ from $\mathbf{w}^{(t)}$ by going a step of size η with the gradient.

$\Rightarrow$ FTRL becomes the GIGA algorithm! ■

7.4 Zero-Sum Games

We consider a **two-player zero-sum games**. These are competitive optimization scenarios with two agents `I` and `II`.

- `I` has k **strategies**, `II` has ℓ strategies. Every pair $(i, j) \in [k] \times [\ell]$ is a **state**.
- Each state has **cost** $a_{ij} \in \mathbb{R}$ to agent `I`. Cost to `II` is $-a_{ij}$, so sum of costs is always 0.
- We think of a matrix $\mathbf{A} = (a_{ij}) \in \mathbb{R}^{k \times \ell}$, where `I` picks a row and `II` a column.
- `I` wants to pick a row to minimize his cost.
- `II` wants to pick a column to maximize cost of `I`.
- Both `I` and `II` can play **randomized strategies**: `I` chooses a distribution $\mathbf{x}$ over the k rows, `II` chooses a distribution $\mathbf{y}$ over the ℓ columns.

Given choices $\mathbf{x}$ and $\mathbf{y}$, it is easy to see that the expected cost for `I` (= profit for `II`) is

$$\sum_{i \in [k]} \sum_{j \in [\ell]} x_i y_j \cdot a_{ij} \;=\; \mathbf{x}^T \mathbf{A} \mathbf{y} \;.$$

Example 17. A popular example is the Rock-Paper-Scissors game. Here both players *simultaneously* choose from $\{R, P, S\}$. The cost becomes

	R	P	S
R	0	-1	1
P	1	0	-1
S	0	1	0

where cost -1 implies that `I` wins, 1 that `I` loses, and 0 indicates a draw. What are the distributions $\mathbf{x}$ and $\mathbf{y}$ that the players would choose here? ■

More fundamentally, whenever an agent optimizes against an adversary, a two-player zero-sum game arises. Notably, worst-case analysis of algorithms for a problem $\mathcal{P}$ can be interpreted as such a game. Consider, e.g., online optimization discussed earlier in this course:

- As a designer (player `I`), we want to choose an algorithm for $\mathcal{P}$ that minimizes the competitive ratio.
- To show that the guarantee holds for all instances, we can imagine a malicious adversary (player `II`) who tries to construct an instance of $\mathcal{P}$ to make the ratio of the chosen algorithm as large as possible.
- Here the strategies of `I` are deterministic algorithms, and a randomized strategy corresponds to a randomized algorithm.
- The randomized strategies of player `II` are all distributions over instances of $\mathcal{P}$.

Consider two **sequential versions** of this game:

(a) First player `I` publicly announces $\mathbf{x}$. Then `II` can answer with the best $\mathbf{y}$.

Here `I` minimizes his cost taking into account the best response of `II`. The optimal expected cost of `I` becomes

$$\min_{\mathbf{x}} \left(\max_{\mathbf{y}} \mathbf{x}^\top \mathbf{A} \mathbf{y} \right)$$

(b) First player II publicly announces $\mathbf{y}$. Then II can answer with the best $\mathbf{x}$.

Here II maximizes the cost of I taking into account the best response of I. The optimal expected cost of I becomes

$$\max_{\mathbf{y}} \left(\min_{\mathbf{x}} \mathbf{x}^\top \mathbf{A} \mathbf{y} \right)$$

Example 18. Consider a game given by the following matrix

$$A = \begin{pmatrix} 0 & 2 & 4 \\ 1 & 0 & -1 \end{pmatrix}$$

In version (a):

- I could announce $(1, 0)$ or $(0, 1)$ which after best response by II would result in cost 4 or 1, respectively.
- The best choice is $\mathbf{x}^* = (\frac{1}{3}, \frac{2}{3})$, since then the expected cost of every column is $x_2 = 2x_1 = 4x_1 - x_2 = \frac{2}{3}$. Hence, the expected cost of I is $\frac{2}{3}$.

In version (b):

- II could announce $(1, 0, 0)$, $(0, 1, 0)$ or $(0, 0, 1)$ which after best response by II would result in cost 0, 0, or -1 for I, respectively.
- The best choice is any vector $\mathbf{y}^* = (\frac{2}{3} + y_3, \frac{1}{3} - 2y_3, y_3)$ with $y_3 \in [0, \frac{1}{6}]$. Then the expected cost of both rows is $2y_2 + 4y_3 = y_1 - y_3 = \frac{2}{3}$. Expected cost of I is again $\frac{2}{3}$.

■

The example looks peculiar – intuitively, we would expect that version (b) is better for I, since he can tailor his answer towards an optimal choice in hindsight. However, the famous Minimax-Theorem shows that, when being played optimally by both players, both variants yield **the same expected cost** for I (and, hence, also for II).

Theorem 39 (Minimax-Theorem)**.** *In every two-player zero-sum game*

$$\min_{\mathbf{x}} \left(\max_{\mathbf{y}} \mathbf{x}^\top \mathbf{A} \mathbf{y} \right) = \max_{\mathbf{y}} \left(\min_{\mathbf{x}} \mathbf{x}^\top \mathbf{A} \mathbf{y} \right) \quad .$$

Recall the interpretation of online optimization as a game. Using this interpretation, it is easy to see that the Minimax-Theorem implies Yao's Principle (Theorem 14).

We will prove the Minimax-Theorem using no-regret learning!

Proof. We first simplify the cost matrix $\mathbf{A}$:

- Subtract $a^- = \min_{i,j} a_{ij}$ from every entry of $\mathbf{A}$. Then $\mathbf{A} \in [0, \rho]^{k \times \ell}$ for some $\rho \geq 0$.
- This changes the expected cost $\mathbf{x}^\top \mathbf{A} \mathbf{y}$ of I by exactly a^- for every $\mathbf{x}$ and $\mathbf{y}$.
- The simplification has no effect on the statement of the theorem.

Consider a repeated version of the game, where in each round $t = 1, \ldots, T$ the same game is played in version (a), i.e., I always moves first. We interpret this as an instance of Online Convex Optimization:

- $D = \{\mathbf{x} \mid \sum_{i=1}^k x_i = 1, x_i \in [0, 1] \text{ for all } i \in [k]\}$, i.e., all randomized strategies of I.

- For each $\mathbf{x}^{(t)} \in D$, the cost becomes

$$c_t(\mathbf{x}^{(t)}) \;=\; c(\mathbf{x}^{(t)}, \mathbf{y}^{(t)}) \;=\; (\mathbf{x}^{(t)})^\top \mathbf{A}(\mathbf{y}^{(t)}),$$

  for a best response $\mathbf{y}^{(t)}$ of `II` against $\mathbf{x}^{(t)}$.
- This is the Experts problem, where each expert is a row $i \in [k]$ and experiences cost $\ell_i^{(t)} = \sum_{j\in[\ell]} a_{ij} y_j^{(t)} \in [0, \rho]$.
- Recall Example 12: c_t satisfies the Lipschitz condition with $L = \rho$ for the ℓ_1-norm. Entropical regularizer is $1/\eta$-strongly convex w.r.t. ℓ_1-norm.
- Hence, if `I` uses FTRL with Entropical regularizer and $\eta = \Theta(1/\sqrt{T})$ to make his strategy choices $\mathbf{x}^{(t)} \in D$, then Theorem 38 implies for the regret

$$\text{Regret}(T) = \sum_{t=1}^{T} c(\mathbf{x}^{(t)}, \mathbf{y}^{(t)}) - \min_{\mathbf{x}\in D} \sum_{t=1}^{T} c(\mathbf{x}, \mathbf{y}^{(t)}) \;\le\; \frac{\ln n}{\eta} + T\rho^2\eta \;=\; \varepsilon(T) \in o(T).$$

For the cost of `I` in round t we observe that, since `II` plays a best response,

$$c(\mathbf{x}^{(t)}, \mathbf{y}^{(t)}) \;=\; \max_{\mathbf{y}} (\mathbf{x}^{(t)})^\top \mathbf{A}\mathbf{y} \;\ge\; \min_{\mathbf{x}} \left(\max_{\mathbf{y}} \mathbf{x}^\top \mathbf{A}\mathbf{y} \right) \;.$$

Consider the average cost of any fixed $\mathbf{x}$ in hindsight. By linearity, this is the same as in a single-round game if `II` would always play an **average history** $\bar{\mathbf{y}}$:

$$\min_{\mathbf{x}\in D} \sum_{t=1}^{T} c(\mathbf{x}, \mathbf{y}^{(t)}) = \min_{\mathbf{x}\in D} \sum_{t=1}^{T} \sum_{i\in[k]} \sum_{j\in[\ell]} x_i a_{ij} y_j^{(t)} = \min_{\mathbf{x}\in D} T \sum_{i\in[k]} \sum_{j\in[\ell]} x_i a_{ij} \sum_{t=1}^{T} \frac{y_j^{(t)}}{T} = T \min_{\mathbf{x}\in D} \mathbf{x}^\top \mathbf{A}\bar{\mathbf{y}}$$

But this is **version (b)** of the problem: `II` first specifies $\bar{\mathbf{y}}$ and then `I` minimizes by choosing $\mathbf{x}$ based on it! Hence, we have that

$$\min_{\mathbf{x}} \mathbf{x}^\top \mathbf{A}\bar{\mathbf{y}} \;\le\; \max_{\mathbf{y}} \left(\min_{\mathbf{x}} \mathbf{x}^\top \mathbf{A}\mathbf{y} \right).$$

Overall,

$$\min_{\mathbf{x}} \left(\max_{\mathbf{y}} \mathbf{x}^\top \mathbf{A}\mathbf{y} \right) - \max_{\mathbf{y}} \left(\min_{\mathbf{x}} \mathbf{x}^\top \mathbf{A}\mathbf{y} \right) \le \frac{1}{T} \cdot \text{Regret}(T) \;\le\; \frac{\varepsilon(T)}{T} \in o(1).$$

Hence, since there is no dependence on T in the first terms, growing $T \to \infty$ implies

$$\min_{\mathbf{x}} \left(\max_{\mathbf{y}} \mathbf{x}^\top \mathbf{A}\bar{\mathbf{y}} \right) \le \max_{\mathbf{y}} \left(\min_{\mathbf{x}} \mathbf{x}^\top \mathbf{A}\mathbf{y} \right) \;.$$

For the proof of the other direction

$$\min_{\mathbf{x}} \left(\max_{\mathbf{y}} \mathbf{x}^\top \mathbf{A}\mathbf{y} \right) \ge \max_{\mathbf{y}} \left(\min_{\mathbf{x}} \mathbf{x}^\top \mathbf{A}\mathbf{y} \right)$$

we define a pair of optimal strategies

$$\mathbf{x}^* \in \arg\min_{\mathbf{x}} \left(\max_{\mathbf{y}} \mathbf{x}^\top \mathbf{A}\mathbf{y} \right)$$
$$\mathbf{y}^* \in \arg\max_{\mathbf{y}} \left(\min_{\mathbf{x}} \mathbf{x}^\top \mathbf{A}\mathbf{y} \right)$$

and observe

$$\begin{aligned} \min_{\mathbf{x}} \left(\max_{\mathbf{y}} \mathbf{x}^\top \mathbf{A}\mathbf{y} \right) &= \max_{\mathbf{y}} \, (\mathbf{x}^*)^\top \mathbf{A}\mathbf{y} \\ &\geq (\mathbf{x}^*)^\top \mathbf{A}\mathbf{y}^* \\ &\geq \min_{\mathbf{x}} \mathbf{x}^\top \mathbf{A}\mathbf{y}^* \\ &= \max_{\mathbf{y}} \left(\min_{\mathbf{x}} \mathbf{x}^\top \mathbf{A}\mathbf{y} \right). \end{aligned} \tag{7.4}$$

Due to the other direction proved above, all inequalities in (7.4) must hold with equality, for any pair of optimal strategies $\mathbf{x}^*$ and $\mathbf{y}^*$. If `I` plays $\mathbf{x}^*$, then $\mathbf{y}^*$ is a best response for `II`. Similarly, if `II` plays $\mathbf{y}^*$, then $\mathbf{x}^*$ must be a best response for `I`. Hence, any pair of optimal strategies is a so-called Nash equilibrium of the game, a collection of mutual best responses. □

Appendix A

Stochastic Concepts and Tools

A.1 Distributions, Conditional and Independent Events

Definition 1. *Let Ω be a finite or countably infinite set (the* sample space*). A* probability distribution $\mathcal{D}$ *over Ω is given by a function*

$$\Pr_{\mathcal{D}} : \Omega \to [0,1] \qquad \textit{such that} \qquad \sum_{\omega \in \Omega} \Pr_{\mathcal{D}}[\omega] = 1.$$

An element ω is called an elementary event*, a subset $A \subset \Omega$ is called an* event*. We use the notation*

$$\Pr_{\mathcal{D}}[A] = \sum_{\omega \in A} \Pr_{\mathcal{D}}[\omega]$$

We often drop the subscript $\mathcal{D}$ when clear from context.

Consider, e.g., outcomes in roulette.

- Elementary events are colors $\Omega = \{\text{black, red, green}\}$.
- $\Pr[\text{black}] = 18/37$, $\Pr[\text{red}] = 18/37$, $\Pr[\text{green}] = 1/37$.
- Elementary events are not always uniformly distributed – if so, we say that we have a **uniform distribution**.
- For the same experiment we can define different elementary events, e.g., we could also choose the numbers $\Omega' = \{0, \ldots, 36\}$. Then we would have a uniform distribution.

Consider two events A and B.

- What is the probability that both happen simultaneously?
- What is the probability that at least one of them happens?
- Clearly: $\Pr[A \wedge B] = \Pr[A \cap B]$ and $\Pr[A \vee B] = \Pr[A \cup B]$.
- We use logical operators "and" ($\wedge$) and "or" ($\vee$) instead of the intersection of sets.
- Similarly as for sets, we have $\Pr[A \vee B] = \Pr[A] + \Pr[B] - \Pr[A \wedge B]$.

When is $\Pr[A \vee B] = \Pr[A] + \Pr[B]$? Only if $A \cap B = \emptyset$, i.e., no overlap in the events.

Is it always true that $\Pr[A \wedge B] = \Pr[A] \cdot \Pr[B]$? No! However, this condition is important, it characterizes **independent events**.

Definition 2. *Events A and B are called* independent *if and only if*

$$\Pr[A \wedge B] = \Pr[A] \cdot \Pr[B] \ .$$

[Pic: Events as sets of outcomes, independent events]

Consider, for exmple, two throws of a coin.

- Two events: A both coins show the same, and B the second throw is heads.
- $\Pr[A \vee B] = 3/4$, $\Pr[A] = 1/2$, and $\Pr[B] = 1/2$.
- Now $\Pr[A \wedge B] = \Pr[A] + \Pr[B] - \Pr[A \vee B] = 1/4 = \Pr[A] \cdot \Pr[B]$
- The events are independent.

As another example, consider a round of roulette.

- Two events: R outcome is red, and U outcome is odd number.
- $\Pr[R] = 18/37 < 1/2$, $\Pr[U] = 18/37 < 1/2$ and $\Pr[RU] = 9/37$.
- Note: $\Pr[R] \cdot \Pr[U] < 18/37 \cdot 1/2 = 9/37 = \Pr[R \wedge U]$.
- The events are not independent.

In the latter example, suppose I bet on odd numbers. Now I see that the outcome is red. Does this information change my chances?

- Initially, I have $\Pr[R] = 18/37$.
- Given that it must be a red number, the 0 and all black numbers have probability 0.
- Among red numbers, there are 9 odd ones out of 18 ones in total.
- Thus, the **conditional probability** that there is an outcome from U given that the outcome is from R is $\Pr[U \mid R] = 9/18 > 18/37$.

Definition 3. $\Pr[U \mid R]$ *is called the* conditional probability *of event U w.r.t. event R.*

[Pic: conditional events, conditional probability]

How do we determine a conditional probability? The set of outcomes shrinks to the ones from R. Thus, we need the probability of outcomes from $R \wedge U$ and normalize it with the one from R. More formally,

$$\Pr[U \mid R] = \frac{\Pr[U \wedge R]}{\Pr[R]} \ .$$

Consider this for two independent events:

$$\Pr[A \mid B] = \frac{\Pr[A \wedge B]}{\Pr[B]} = \frac{\Pr[A] \cdot \Pr[B]}{\Pr[B]} = \Pr[A] \ .$$

Observation 1. $\Pr[A \mid B] = \Pr[A]$ *if and only if A and B are independent events.*

Consider the following experiment.

- We throw a dice two times.
- "≥ 10" is the event that the numbers sum to at least 10.
- "First $= x$" is the event that the first throw shows the number x, where $x = 1, \ldots, 6$.

We can formulate the $\Pr[\geq 10]$ by exploring all possibilities for First:

$$\begin{aligned}\Pr[\geq 10] &= \Pr[\geq 10 \wedge \text{First} = 1] + \Pr[\geq 10 \wedge \text{First} = 2] + \ldots + \Pr[\geq 10 \wedge \text{First} = 6] \\ &= \Pr[\text{First} = 1] \cdot \Pr[\geq 10 \mid \text{First} = 1] + \Pr[\text{First} = 2] \cdot \Pr[\geq 10 \mid \text{First} = 2] \\ &\quad + \ldots + \Pr[\text{First} = 6] \cdot \Pr[\geq 10 \mid \text{First} = 6] \\ &= 0 + 0 + 0 + \frac{1}{36} + \frac{2}{36} + \frac{3}{36} = \frac{6}{36}\end{aligned}$$

Why does this hold? The main property is that the collection of "First" events are mutually disjoint and cover the whole sample space of possible outcomes.

[Pic: Partition of sample space, event sliced up by partition]

Definition 4. *Suppose events $B_1 \ldots, B_k$ are a* partition of the sample space*, i.e., $B_i \cap B_j = \emptyset$ if $i \neq j$ and $\bigcup_i B_i = \Omega$. Then the* law of total probability *states that for every event $A \subseteq \Omega$*

$$\Pr[A] = \sum_{i=1}^{k} \Pr[B_i] \cdot \Pr[A \mid B_i].$$

Note that we used this formula, e.g., in the proof of Theorem 7. The events A_t (best person arrives in round t) are mututally disjoint events that cover the whole range of possible outcomes. As such the probability of accepting the best person could be experssed as

$$\sum_{t=s+1}^{n} \Pr[A_t] \cdot \Pr[R_{(1,t-1)} \mid A_t].$$

A.2 Random Variables, Expectation, Concentration

The value of the outcome of an experiment is expressed using **random variables**.

- Suppose you earn 9€ if the sum of numbers in two throws of a dice is ≥ 10.
- However, we must pay 1.5€ when the sum is < 10.
- Should we play this game?
- The number of interest here is the **expectation** or **expected value**, which is also the average reward when the game is repeated very often.
- Recall $\Pr[\geq 10] = 6/36 = 1/6$, so $\Pr[< 10] = 5/6$.
- Hence, the expected reward is $9 \cdot \frac{1}{6} - 1.5 \cdot \frac{5}{6} = \frac{1}{4}$.
- Indeed, in the long run it pays off to play the game.

To express this scenario more formally, we use a random variable X for the assignment of payoffs (9 or -1.5€) to the set of elementary events of the experiment, and the expected value of X to express the average reward weighted by the probabilities of the elementary events.

Definition 5. *A* random variable *is a function $X : \Omega \to \mathbb{R}$.*

An important class of random variables are indicator variables.

Definition 6. $X \in \{0,1\}$ *is called* indicator *or* Bernoulli variable. *The variable corresponds to event* $A_X = \{\omega \mid X(\omega) = 1\} \subseteq \Omega$ *and indicates whether the event has occured or not.*

Let us define the expected value.

Definition 7. *The* expected value $\mathbb{E}[X]$ *of random variable* X *is*

$$\mathbb{E}[X] = \sum_{\omega \in \Omega} X(\omega) \cdot \Pr[\omega] = \sum_{r \in \mathbb{R}} r \cdot \Pr[X = r]$$

For an indicator variable X *we have* $\mathbb{E}[X] = \sum_{\omega \in A_X} 1 \cdot \Pr[\omega] = \Pr[X = 1] = \Pr[A_X]$.

Some observations:

- X is not an event, so $\Pr[X]$ makes no sense!
- Events based on random variables have the form $\Pr[X = 0]$, $\Pr[X \geq 15]$, $\Pr[X > 3 \wedge X \neq 7]$ etc.
- Since Ω is finite or countably inifinite, the number of possible values r such that there is ω with $X(\omega) = r$ is also finite or countably infinite. Hence, $\sum_{r \in \mathbb{R}} r \cdot \Pr[X = r]$ is well-defined.
- If X and Y are random variables, then $X + Y$, $X \cdot Y$ or arbitrary functions $f(X)$ or $f(X, Y)$ are also random variables.

A very important property is linearity of expectation, i.e., if we apply a linear transformation to X and Y, the expected value also changes linearly. This is a direct consequence of the fact that the expected value is a weighted average of random values and probabilities.

Theorem 40. *The* expected value *is a* linear function. *For any* $\lambda \in \mathbb{R}$ *and any two random variables* X *and* Y

$$\begin{aligned} \mathbb{E}[X + Y] &= \mathbb{E}[X] + \mathbb{E}[Y] \\ \mathbb{E}[\lambda \cdot X] &= \lambda \cdot \mathbb{E}[X] \\ \mathbb{E}[X + \lambda] &= \mathbb{E}[X] + \lambda \end{aligned}$$

Proof. We only prove the first statement. The other ones can be shown similarly.

$$\begin{aligned} \mathbb{E}[X] + \mathbb{E}[Y] &= \sum_{x \in \mathbb{R}} x \cdot \Pr[X = x] + \sum_{y \in \mathbb{R}} y \cdot \Pr[Y = y] \\ &= \sum_{x \in \mathbb{R}} \sum_{y \in \mathbb{R}} x \cdot \Pr[X = x] \cdot \Pr[Y = y \mid X = x] \\ &\quad + \sum_{y \in \mathbb{R}} \sum_{x \in \mathbb{R}} y \cdot \Pr[Y = y] \cdot \Pr[X = X \mid Y = y] \\ &= \sum_{x \in \mathbb{R}} \sum_{y \in \mathbb{R}} x \cdot \Pr[(X, Y) = (x, y)] + \sum_{y \in \mathbb{R}} \sum_{x \in \mathbb{R}} y \cdot \Pr[(X, Y) = (x, y)] \\ &= \sum_{x \in \mathbb{R}, y \in \mathbb{R}} (x + y) \cdot \Pr[(X, Y) = (x, y)] \\ &= \mathbb{E}[X + Y] \end{aligned}$$

□

Can we use the expectation of X to infer anything about the probability that $X \geq a$ or $X \leq a$ for some fixed value $a \in \mathbb{R}$? Vaguely speaking, if the expectation of X is "large", does this mean that X must also be "large" with "high" probability? Indeed, if the random variable is non-negative, formally precise statements of this kind can be derived.

Theorem 41. *Let $X \geq 0$ be a* non-negative *random variable and $a \geq \mathbb{E}[X]$ a number. Then the* Markov inequality *holds:*

$$\Pr[X \geq a] \quad \leq \quad \frac{\mathbb{E}[X]}{a} \ .$$

Proof. A direct calculation:

$$\begin{aligned}
\mathbb{E}[X] &= \sum_r r \cdot \Pr[X = r] \\
&= \sum_{r \geq a} r \cdot \Pr[X = r] + \sum_{r < a} r \cdot \Pr[X = r] \\
&\geq \sum_{r \geq a} r \cdot \Pr[X = r] \\
&\geq a \cdot \sum_{r \geq a} \Pr[X = r] = a \cdot \Pr[X \geq a]
\end{aligned}$$

□

A much stronger and very useful bound can be shown for a collection of independent Bernoulli variables.

Theorem 42. *Let $X_1, \ldots, X_n$ be independent Bernoulli variables with $\mathbb{E}[X_i] = \Pr[X_i = 1] = p_i$, and let $X = \sum_{i=1}^n X_i$. Then the following* Chernoff inequalities *hold:*

1. *For every $\delta > 0$*

$$\Pr[X \geq (1+\delta) \cdot \mathbb{E}[X]] \quad \leq \quad \left(\frac{e^\delta}{(1+\delta)^{1+\delta}} \right)^{\mathbb{E}[X]} \quad \leq \quad e^{-\mathbb{E}[X] \cdot \delta^2/3}$$

2. *For $0 < \delta < 1$*

$$\Pr[X \leq (1-\delta) \cdot \mathbb{E}[X]] \quad \leq \quad \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}} \right)^{\mathbb{E}[X]} \quad \leq \quad e^{-\mathbb{E}[X] \cdot \delta^2/2}$$

For the proof, we can apply Markov inequalities to the random variable $e^{\alpha X}$ for a suitably chosen $\alpha = \ln(1+\delta)$.

[Pic: concentration around expectation]