

## The Minimax Theorem and Online Learning

Notation:

- $m \times n$  payoff matrix  $\mathbf{A}$ — $a_{ij}$  is the row player's payoff for outcome  $(i, j)$  when row player plays strategy  $i$  and column player plays strategy  $j$
- mixed row strategy  $\mathbf{x}$  (a distribution over rows)
- mixed column strategy  $\mathbf{y}$  (a distribution over columns)

Expected payoff of the row player:

$$\begin{aligned} \sum_{i=1}^m \sum_{j=1}^n \Pr[\text{outcome } (i, j)] a_{ij} &= \sum_{i=1}^m \sum_{j=1}^n \underbrace{\Pr[\text{row } i \text{ chosen}]}_{=x_i} \underbrace{\Pr[\text{column } j \text{ chosen}]}_{=y_j} a_{ij} \\ &= \mathbf{x}^T \mathbf{A} \mathbf{y} \end{aligned}$$

**Theorem 1** (Minimax Theorem). *For every two-player zero-sum game  $\mathbf{A}$ ,*

$$\max_{\mathbf{x}} \left( \min_{\mathbf{y}} \mathbf{x}^T \mathbf{A} \mathbf{y} \right) = \min_{\mathbf{y}} \left( \max_{\mathbf{x}} \mathbf{x}^T \mathbf{A} \mathbf{y} \right). \quad (1)$$

Or in English, the expected payoff of the row player is the same whether the row player goes first or second. This is called the *value of the game*.

## From LP Duality to Minimax

$$\max_{\mathbf{x}} \left( \min_{\mathbf{y}} \mathbf{x}^T \mathbf{A} \mathbf{y} \right) = \max_{\mathbf{x}} \left( \min_{j=1}^n \mathbf{x}^T \mathbf{A} \mathbf{e}_j \right) \quad (2)$$

$$= \max_{\mathbf{x}} \left( \min_{j=1}^n \sum_{i=1}^m a_{ij} x_i \right) \quad (3)$$

where  $\mathbf{e}_j$  is the  $j^{\text{th}}$  standard basis vector:

$$(\mathbf{e}_j)_i = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

max  $v$

subject to

$$v - \sum_{i=1}^m a_{ij} x_i \leq 0 \quad \text{for all } j = 1, \dots, n$$

$$\sum_{i=1}^m x_i = 1$$

$$x_1, \dots, x_m \geq 0 \quad \text{and} \quad v \in \mathbb{R}.$$

min  $w$

subject to

$$w - \sum_{j=1}^n a_{ij} y_j \geq 0 \quad \text{for all } i = 1, \dots, m$$

$$\sum_{j=1}^n y_j = 1$$

$$y_1, \dots, y_n \geq 0 \quad \text{and} \quad w \in \mathbb{R}.$$

## Online Learning

So far: we assumed we could see the future (e.g., scheduling, caching).

What if we can't see the future? This is called an *online* setting, not like the internet, but as if the input is waiting *on line*.

### An Online Problem

1. The input arrives “one piece at a time.”
2. An algorithm makes an irrevocable decision each time it receives a new piece of the input.

## Online Decision Making

Choose from expert predictions every time step. An adversary decides who predicts well/poorly in response to your strategy.

### Online Decision-Making

At each time step  $t = 1, 2, \dots, T$  :

a decision-maker picks a probability distribution  $\mathbf{p}^t$  over her experts or actions  $i = 1, \dots, N$

an adversary picks a **loss** vector  $\ell^t : A \rightarrow [0, 1]$

an action  $i^t$  is chosen according to the distribution  $\mathbf{p}^t$ , and the decision-maker receives loss  $\ell_i^t$

the decision-maker learns  $\ell^t$ , the entire **loss** vector

The input arrives “one piece at a time.”

## What should we compare to?

Thus far, we've been trying to achieve optimal solutions, or comparing to optimal solutions assuming we know full information about the future and what is optimal. Does that still make sense?

**Example 1** (Comparing to the Best Action Sequence). Suppose your set of experts (or actions) is  $A = \{1, 2\}$ . Each day  $t$ , the adversary chooses the loss vector  $\ell^t$  as follows: if the algorithm chooses a distribution  $\mathbf{p}^t$  for which the probability on action 1 is at least  $\frac{1}{2}$ , then  $\ell^t$  is set to the vector  $(1, 0)$ . Otherwise, the adversary sets  $\ell^t$  equal to  $(0, 1)$ .

This adversary forces the expected cumulative loss of the algorithm to be at least

Thus, the algorithm's approximation is

**Definition 1** (Regret). Fix loss<sup>1</sup> vectors  $\ell^1, \dots, \ell^T$ . The regret of the action sequence  $a^1, \dots, a^T$  is

$$\underbrace{\sum_{t=1}^T \ell^t(a^t)}_{\text{our algorithm}} - \underbrace{\min_{i=1}^N \sum_{t=1}^T \ell_i^t}_{\text{best fixed action}} . \quad (4)$$

Specifically, our goal is to minimize *regret*.

**Example 2** (Randomization Is Necessary for No Regret). Fix a deterministic online decision-making algorithm. At each time step  $t$ , the algorithm commits to a single action  $a_t$ . The obvious strategy for the adversary is to set the loss of action  $a_t$  to 1, and the loss of every other action to 0. Then, the cumulative loss of the algorithm is

while the cumulative loss of the best action in hindsight is at most

So the regret of any deterministic algorithm is at least

Even when there are only 2 actions, for arbitrarily large  $T$ , the worst-case regret of the algorithm is at least

---

<sup>1</sup>Note that for the rewards setting, the definition of regret would instead be  $\max_{i=1}^N \sum_{t=1}^T r_i^t - \sum_{t=1}^T r^t(a^t)$ , still minimizing the difference between the algorithm and the best fixed action, but now the maximum reward for the best fixed action will be larger than the algorithm instead of the minimum loss being smaller.

**Example 3** (Regret Lower Bound). Suppose there are  $n = 2$  actions, and that we choose each loss vector  $\ell^t$  independently and equally likely to be  $(0, 1)$  or  $(1, 0)$ . No matter how smart or dumb an online decision-making algorithm is, with respect to this random choice of loss vectors, its expected loss at each time step is exactly

and its expected cumulative loss is thus

The expected cumulative loss of the best fixed action in hindsight is

This follows from the fact that if a fair coin is flipped  $T$  times, then the expected number of heads is  $\frac{T}{2}$  and the standard deviation is  $\frac{1}{2}\sqrt{T}$ .

What if there are  $n$  actions? A similar argument shows that, with  $n$  actions, the expected regret of an online decision-making algorithm cannot grow more slowly than  $b\sqrt{T \ln n}$ , where  $b > 0$  is some constant independent of  $n$  and  $T$ .