ECE 8803: Online Decision Making in Machine Learning **Fall 2021**

Lecture 25: November 29

Lecturer: Vidya Muthukumar

In this course, we have discussed various paradigms and algorithms for decision-making primarily from the perspective of a *single agent*, who wishes to maximize reward or minimize loss. However, in many modern decision-making settings, agents do not act in isolation: instead, they learn from each other and influence each other's behavior. When we consider such multi-agent interaction, two natural questions arise:

- What is the *ideal* outcome of interaction for each agent?
- Which learning algorithms, if any, lead to this ideal outcome?

Our last two lectures will explore answers to these questions. The first question is in the purview of *game theory*, a field that originated in mathematics and economics and formalizes notions of optimality in multi-agent interaction. We will introduce basic concepts in game theory in today's and Wednesday's lecture. To answer the second question, we will revisit the very first algorithms that we introduced in this class—that minimized regret in adversarial online prediction.

25.1. Examples of strategic (adversarial) interaction

In today's lecture, we will especially focus on *adversarial* interaction betweem two agents, where one agent wants to maximize reward and the other wants to minimize reward. Before formally introducing game-theoretic concepts, we will colloquially discuss a few examples in which such adversarial interaction naturally arises.

Example 1 *[RL for game-playing] Some of the biggest success stories in reinforcement learning have been in* ***game playing****, where the goal of an RL agent is to win a game against her opponent. For example, Figure 25.1 depicts the complex game of Go, and the RL agent AlphaGo has achieved impressive successes against human champions in the last few years. How do we define an optimal strategy for AlphaGo; in particular, can we use the Markov Decision Process theory that we developed in the last few weeks? The answer to this turns out to not be quite so simple: the evolution of the state of the game depends not only on the actions that AlphaGo will take,* ***but also the actions of her human opponent****. Therefore, the optimal course of action for AlphaGo will intricately depend on how her human opponent behaves. In an idealized world, an optimal strategy for AlphaGo*

Figure 25.1: Using RL to solve the game of Go.

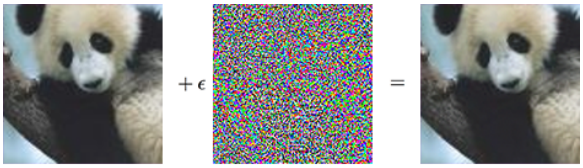

Figure 25.2: The adversarial examples problem in ML.

would be one that maximizes her chances of winning the game against the worst case: a ***perfect human adversary***[1]*.*

Example 2 *[Adversarial training in ML] A second example involving strategic interaction is the example of* ***adversarial examples*** *in ML. ML algorithms are trained to handle stochastic data, and indeed perform very well on uncorrupted test datasets. However, this performance is known to not be robust: as depicted in Figure 25.2, an infinitesmal perturbation of an input can lead to a wildly wrong prediction. This lack of robustness has critical implications for security-related applications in AI: in particular, such adversarial examples can easily fool spam and virus detectors. Solving the adversarial examples problem consequently becomes a* ***zero-sum game*** *between machine learner and attacker. The machine learner wishes to minimize prediction errors (equivalently, accurately detect spam), while the adversary wishes to maximize prediction errors (by minimizing the detection of spam).*

25.2. Fundamentals of zero-sum game theory

With our motivation for studying adversarial interaction clear, we now introduce a formal framework for studying zero-sum games. We first focus on the case where the decision set of the two agents is finite, and define such finite zero-sum games below.

1. Note that humans are also not perfect decision-makers: in a certain sense, AlphaGo is still optimized using single-agent RL principles to maximize her chances of winning against *known models* for expert humans. This helps us avoid the game-theoretic conundrum. Whether AlphaGo achieves te idealized objective of maximizing performance against a mythical perfect adversary is in fact an open question: an intriguing question is what would happen if two versions of AlphaGo were pitted against each other?

	Heads (P2)	Tails (P2)
Heads (P1)	1	0
Tails (P1)	0	1

Definition 1 (Finite zero-sum game) *A two player, finite zero-sum game is defined by action spaces $\mathcal{A}_1 := \{1, \ldots, m\}$ and $\mathcal{A}_2 := \{1, \ldots, n\}$ for players 1 and 2 respectively, and common "payoff matrix" $\boldsymbol{A} := \{A_{ij}\}_{(i,j)\in\mathcal{A}_1\times\mathcal{A}_2}$ whose interpretation is as follows: when player 1 plays action i and player 2 plays action j, player 1 obtains reward equal to A_{ij} and player 2 obtains reward equal to $-A_{ij}$. Equivalently, player 1 wants to **maximize** the payoff function specified by $\boldsymbol{A}$, while player 2 wants to **minimize** this payoff function. The action tuple (i, j) that is picked by players 1 and 2 is commonly called a **pure strategy pair**. Players can also mix their pure strategies in a probability distribution, which we will examine subsequently.*

It is worth noting from the above definition that regardless of the action tuple (i, j) that is played, the *sum* of the rewards of both players is equal to $A_{ij} - A_{ij} = 0$. This is the reason for the moniker "zero-sum game". An equivalent representation of a zero-sum game is a *constant-sum game*, in which there are now *two* payoff matrices $\boldsymbol{A}, \boldsymbol{B}$ such that player 1 and player 2 receive rewards A_{ij}, B_{ij} respectively, and regardless of the action tuple (i, j) we have $A_{ij} + B_{ij} = c$ for some constant c. Note that the constant-sum constraint ensures that a relative gain for player 1 constitutes a relative loss for player 2, and vice-versa—therefore, it maintains the adversarial nature of interaction. In fact, a constant-sum game is only a shifted version of a zero-sum game, and so the eventual outcome and recommended strategy for players does not change with this shift. Nevertheless, the constant-sum interpretation is useful as it is often easier to connect to practical scenarios of adversarial interaction. The game-playing Example 1 in particular is more easily captured by a constant-sum framework: c points are up for grabs on every round, and are split across the two players.

Zero-sum games as defined in Definition 1 have a concise representation as a $m \times n$ *payoff table*, i.e. the payoff matrix $\boldsymbol{A}$ written in tabular form. We describe two popular examples of zero-sum games below. Henceforth, we abbreviate player 1 to P1 and player 2 to P2.

Example 3 *[Matching pennies game] One of the simplest examples of a zero-sum game is the **matching pennies** game. Here, P1 and P2 can each take one out of 2 actions: "Heads (H)" or "Tails (T)", corresponding to a type of coin. P1 wants to **match** the coin given by P2, while P2 wants to **mismatch** the coin given by P1. This is a natural zero-sum interaction, and the payoff table is given in Table 3. The matching pennies game is isomorphic (i.e. equivalent) to the binary sequence prediction game, in which P1's action is the sequence prediction $\widehat{X}_t$ and P2's action is the sequence realization X_t. In adversarial sequence prediction, P1 wants to match the prediction to the realization, while P2 wants to make P1 make as many mistakes as possible.*

Example 4 *[Rock-paper-scissors game] We can take Example 3 a little further to define the rock-paper-scissors game, which is also a zero-sum game. Here, P1 and P2 can each take one out of 3 actions: "Rock (R)", "Paper (P)" or "Scissors (S)". The rules of rock-paper-scissors are simple (and you may be familiar with them if you've played the game yourself): rock*

	Rock (P2)	Paper (P2)	Scissors (P2)
Rock (P1)	0	-1	1
Paper (P1)	1	0	-1
Scissors (P1)	-1	1	0

beats scissors, scissors beats paper, and paper beats rock. The payoff Table 4 is defined appropriately from the point of view of P1.

Examples 3 and 4 are quintessential examples in zero-sum game theory, and provide two simple illustrations of what a payoff table looks like. So far, our discussion of payoffs has centered around so-called *pure strategies*: player 1 plays action i deterministically, and player 2 plays action j deterministically. We will see shortly that it will be useful to allow the players to choose probability distributions over these pure strategies, thus leading to *mixed strategies*. Formally, a mixed-strategy pair is given by a tuple $(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x}$ is a probability distribution over the m actions of P1 and $\boldsymbol{y}$ is a probability distribution over the n actions of P2. In this language, when P1 plays $\boldsymbol{x}$ and P2 plays $\boldsymbol{y}$, P1 receives reward equal to

$$\sum_{i,j} x_i y_j \cdot A_{ij} = \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{y}.$$

Similarly, P2 receives reward equal to $-\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{y}$.

25.2.1 Formalizing optimality in zero-sum games: Nash equilibrium

Now that we have formalized the definition of a zero-sum game, we ask what *optimal strategies* for P1 and P2 look like. Informally speaking, the challenge with multi-agent interaction is that there is no fixed optimal action for a player. For example, in the rock-paper-scissors game in Example 4 none of the pure strategies "Rock", "Paper" or "Scissors" are unequivocally optimal. Moreover, depending on what P2 does, any of these strategies could in fact be optimal. In particular, P1's *best-response* depends on what P2 does in the following ways:

- If P2 plays "Rock", P1's best response is "Paper".
- If P2 plays "Paper", P1's best response is "Scissors".
- If P2 plays "Scissors", P1's best response is "Rock".

This tells us that P1's notion of optimality will depend on P2's notion of optimality, and vice-versa. The right way to formalize optimality in multi-agent interaction turns out to be a concept called *Nash equilibrium*, which is defined below.

Definition 2 *A mixed strategy-pair $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is a Nash equilibrium of a zero-sum game defined by payoff matrix $\boldsymbol{A}$ if we have:*

$$(\boldsymbol{x}^*)^\top \boldsymbol{A}\boldsymbol{y}^* \geq \sum_{j=1}^{n} A_{ij} y_j^* \text{ for all } i = 1, \ldots, m \text{ and}$$

$$(\boldsymbol{x}^*)^\top \boldsymbol{A}\boldsymbol{y}^* \leq \sum_{i=1}^{m} A_{ij} x_i^* \text{ for all } j = 1, \ldots, n.$$

*In other words, P1's mixed strategy $\boldsymbol{x}^*$ is a **best-response** to P2's mixed strategy $\boldsymbol{y}^*$, and P2's mixed strategy $\boldsymbol{y}^*$ is a **best-response** to P1's mixed strategy $\boldsymbol{x}^*$.*

The Nash equilibrium constitutes a notion of optimality in multi-agent interaction because it ensures that neither player, P1 nor P2, wishes to deviate from the outcome $(\boldsymbol{x}^*, \boldsymbol{y}^*)$: in other words, their strategies constitute best responses to each other. Notice, also, that we have defined Nash equilibrium in the context of mixed strategies: this is because a pure-strategy Nash equilibrium need not always exist. To illustrate, let us consider the matching pennies Example 3. Here, (H, H) does not constitute a Nash equilibrium since P2 is incentivized to deviate to T (since he wants to mismatch); similarly, (T, T) would not be a Nash equilibrium either. On the other hand, (H, T) does not constitute a Nash equilibrium either, since P1 is incentivized to deviate to T (since she wants to match); similarly, (T, H) would not be a Nash equilibrium either. *It turns out that the strategy $(\boldsymbol{x}^*, \boldsymbol{y}^*) = (1/2, 1/2)$, where both players choose H and T uniformly at random, is the unique Nash equilibrium of the matching pennies game.* In fact, similar logic can be used to show that the unique Nash equilibrium of the rock-paper-scissors game involves both P1 and P2 randomizing uniformly among the strategies "Rock", "Paper" and "Scissors".

We saw explicit examples of Nash equilibria in the game-playing Examples 3 and 4. How would we compute or find a Nash equilibrium of a larger, more complex game? A starting question is whether we can even guarantee that a Nash equilibrium always exists.

It turns out that a Nash equilibrium indeed always exists[2]. For the case of zero-sum games, this is due to a seminal *minimax theorem* by von Neumann (developed in the 1920's). The minimax theorem (for finite games) is as follows:

Theorem 3 (von Neumann's minimax theorem) *For any payoff matrix $\boldsymbol{A}$, we have*

$$\min_{\boldsymbol{y}} \max_{\boldsymbol{x}} \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{y} = \max_{\boldsymbol{x}} \min_{\boldsymbol{y}} \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{y} =: A^*. \tag{25.1}$$

The number A^* is called the value of the game. Equation (25.1) *ensures that any Nash equilibrium tuple $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ satisfies $(\boldsymbol{x}^*)^\top \boldsymbol{A}\boldsymbol{y}^* = A^*$, i.e. any Nash equilibrium needs to achieve the unique value of the game A^*.*

Theorem 3 essentially implies that the order in which we take a minimum and maximum does not impact the eventual value, and has several generalizations as well as profound implications. The minimax theorem was also generalized to convex-concave games, as we will see in Section 25.3.3 of this lecture note—this has profound implications for min-max optimization and even constrained convex optimization. We do not prove the minimax theorem here, but instead use it as a black box throughout today's lecture. In particular, it is useful to review why the minimax theorem statement (Equation (25.1)) implies that any A^*-achieving tuple is a Nash equilibrium. An equivalent rephrasing of Definition 2 is as follows: $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is a Nash equilibrium if we have

$$(\boldsymbol{x}^*)^\top \boldsymbol{A}\boldsymbol{y}^* = \min_{\boldsymbol{y}} (\boldsymbol{x}^*)^\top \boldsymbol{A}\boldsymbol{y} \text{ and}$$
$$(\boldsymbol{x}^*)^\top \boldsymbol{A}\boldsymbol{y}^* = \max_{\boldsymbol{x}} \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{y}^*.$$

2. This is also the case for non-zero-sum games, as we will see in the last lecture.

From the above, we have

$$\max_{\boldsymbol{x}} \min_{\boldsymbol{y}} \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{y} \leq \max_{\boldsymbol{x}} \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{y}^* = (\boldsymbol{x}^*)^\top \boldsymbol{A}\boldsymbol{y}^* = \min_{\boldsymbol{y}} (\boldsymbol{x}^*)^\top \boldsymbol{A}\boldsymbol{y} \leq \min_{\boldsymbol{y}} \max_{\boldsymbol{x}} \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{y},$$

and so

$$\max_{\boldsymbol{x}} \min_{\boldsymbol{y}} \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{y} \leq (\boldsymbol{x}^*)^\top \boldsymbol{A}\boldsymbol{y}^* \leq \min_{\boldsymbol{y}} \max_{\boldsymbol{x}} \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{y}.$$

However, Equation (25.1) tells us that $\min_{\boldsymbol{y}} \max_{\boldsymbol{x}} \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{y} = \max_{\boldsymbol{x}} \min_{\boldsymbol{y}} \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{y} =: A^*$, which gives us $A^* \leq (\boldsymbol{x}^*)^\top \boldsymbol{A}\boldsymbol{y}^* \leq A^*$! This is only possible if $(\boldsymbol{x}^*)^\top \boldsymbol{A}\boldsymbol{y}^* = A^*$, i.e. the Nash equilibrium tuple $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ achieves the value of the game A^*. Finally, Equation (25.1) also tells us that a Nash equilibrium always exists: for example, we simply compute $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \max_{\boldsymbol{y}} \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{y}$, and look at the corresponding $\boldsymbol{y}^* = \arg\max_{\boldsymbol{y}} (\boldsymbol{x}^*)^\top \boldsymbol{A}\boldsymbol{y}$. (We could also do this the other way round.)

In summary, the profound implication of Theorem 3 is that *in a zero-sum game, achieving equilibrium is equivalent to achieving the equilibrium value A^*.* It is worth keeping this implication in mind as we examine how to approach Nash equilibrium in Section 25.3.

25.3. Approaching Nash equilibrium through regret minimization

Thus far, we have seen that for any finite game (described succinctly through a payoff matrix $\boldsymbol{A}$), a Nash equilibrium $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ always exists. While this is reassuring, we would ideally like to understand how and whether players naturally approach such equilibrium behavior. In particular, do there exist algorithms whereby P1 and P2 learn from each other, and naturally *converge* to Nash equilibrium behavior?

For the case of zero-sum (adversarial) interaction, the answer turns out to be yes—specifically, when players use "no-regret" online learning algorithms of the kind that we examined in the first month of this class, and play them against each other! To see how this is the case, we will first write game-playing in the "experts" framework.

25.3.1 Game-playing in the experts framework

To write game-playing in the "prediction-with-expert-advice" framework, we consider the decision of P1 at round t to be any probability distribution over her m pure strategies, given by $\boldsymbol{x}_t$, and the decision of P2 at round t to be any probability distribution over his n pure strategies, given by $\boldsymbol{y}_t$. The "experts" for P1 correspond to her m pure strategies. At round t, if P2 plays $\boldsymbol{y}_t$, P1 will receive a reward of $g_{1,t}(i) := \sum_{j=1}^n y_{t,j} A_{ij}$ if she chose pure strategy (expert) i. Consequently, P1's reward at round t from choosing the distribution over her strategies (experts) $\boldsymbol{x}_t$ is given by $\sum_{i=1}^m \sum_{j=1}^n x_{t,i} y_{t,j} A_{ij} =: \boldsymbol{x}_t^\top \boldsymbol{A}\boldsymbol{y}_t$.

Similarly, the "experts" for P2 correspond to his n pure strategies. At round t, if P1 plays $\boldsymbol{x}_t$, P2 will receive a reward of $g_{2,t}(j) := -\sum_{i=1}^m x_{t,i} A_{ij}$ if he chose pure strategy (expert) j. Consequently, P2's *reward* at round t from choosing the distribution over his strategies (experts) $\boldsymbol{y}_t$ is given by $-\sum_{j=1}^n \sum_{i=1}^m y_{t,j} x_{t,i} A_{ij} =: -\boldsymbol{x}_t^\top \boldsymbol{A}\boldsymbol{y}_t$. Equivalently, we can think of P2 incurring a *loss* of $\boldsymbol{x}_t^\top \boldsymbol{A}\boldsymbol{y}_t$.

In this framework, the respective goals of P1 and P2 are clear: P1 wishes to *maximize* $\sum_{t=1}^T \boldsymbol{x}_t^\top \boldsymbol{A}\boldsymbol{y}_t$, while P2 wishes to *minimize* $\sum_{t=1}^T \boldsymbol{x}_t^\top \boldsymbol{A}\boldsymbol{y}_t$. For example, in the matching

pennies game, P1 will pick distributions over H and T to maximize the expected number of matches while P2 will pick distributions to try and minimize the number of matches.

25.3.2 No-regret implies convergence to Nash

Writing game-playing in this experts-based framework allows us to define *no-regret* strategies for P1 and P2, which we do below.

Definition 4 *An algorithm used by P1 that generates $\{\boldsymbol{x}_t\}_{t=1}^T$ is **no-regret** if, for any sequence $\{\boldsymbol{y}_t\}_{t=1}^T$ that is generated by P2, we have*

$$\max_{\boldsymbol{x}} \sum_{t=1}^T \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{y}_t - \sum_{t=1}^T \boldsymbol{x}_t^\top \boldsymbol{A}\boldsymbol{y}_t =: R_T = o(T).$$

*That is, P1 achieves, through her algorithm, almost as much reward as she could have had she used the best **fixed** strategy in hindsight. Similarly, an algorithm used by P2 that generates $\{\boldsymbol{y}_t\}_{t=1}^T$ is **no-regret** if, for any sequence $\{\boldsymbol{x}_t\}_{t=1}^T$ that is generated by P1, we have*

$$\sum_{t=1}^T \boldsymbol{x}_t^\top \boldsymbol{A}\boldsymbol{y}_t - \min_{\boldsymbol{y}} \sum_{t=1}^T \boldsymbol{x}_t^\top \boldsymbol{A}\boldsymbol{y} =: R_T = o(T).$$

*That is, P2 achieves, through his algorithm, almost as much reward as he could have had he used the best **fixed** strategy in hindsight.*

Definition 4 should remind you of the definitions of no-regret that we explored early in the class for online prediction and online convex optimization. Then, we explored explicit designs of no-regret algorithms from the point of view of *one* of the agents (either P1 or P2): for example, multiplicative weights and follow-the-perturbed-leader both satisfied the condition in Definition 4. Now, we will ask what happens when these no-regret algorithms are *pitted against each other*: it turns out that the outcome, in a certain sense, is Nash equilibrium behavior. The following result states this formally.

Theorem 5 *Consider any zero-sum game $\boldsymbol{A}$ and assume that P1 and P2 play no-regret algorithms with regret bound given by R_T. Further, consider the average strategy $\overline{\boldsymbol{x}}_T := \frac{1}{T}\sum_{t=1}^T \boldsymbol{x}_t$ and $\overline{\boldsymbol{y}}_T := \frac{1}{T}\sum_{t=1}^T \boldsymbol{y}_t$. We have*

$$|(\overline{\boldsymbol{x}}_T)^\top \boldsymbol{A}\overline{\boldsymbol{y}}_T - A^*| \leq \frac{2R_T}{T}.$$

Theorem 5, together with the minimax theorem, imply that the strategy tuple $(\overline{\boldsymbol{x}}_T, \overline{\boldsymbol{y}}_T)$ is in fact an approximate-Nash equilibrium of the zero-sum game! Moreover, since we have seen that we can design algorithms with $R_T = \mathcal{O}(\sqrt{T})$, Theorem 5 implies that we can get a $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$ rate of convergence to Nash equilibrium.

This result has a short and beautiful proof that we provide below.

Proof First, the no-regret condition for P1 implies that

$$\frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t^\top \boldsymbol{A}\boldsymbol{y}_t \geq \max_{\boldsymbol{x}} \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{y}_t - \frac{R_T}{T}$$

$$\implies \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t^\top \boldsymbol{A}\boldsymbol{y}_t \geq \max_{\boldsymbol{x}} \boldsymbol{x}^\top \boldsymbol{A}\overline{\boldsymbol{y}}_T - \frac{R_T}{T}. \tag{25.2}$$

Similarly, the no-regret condition for P2 implies that

$$\frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t^\top \boldsymbol{A}\boldsymbol{y}_t \leq \min_{\boldsymbol{y}} (\overline{\boldsymbol{x}}_T)^\top \boldsymbol{A}\boldsymbol{y} + \frac{R_T}{T}. \tag{25.3}$$

We will prove Theorem 5 in two steps: first, we will show that the algorithm's average performance, i.e. $\frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t^\top \boldsymbol{A}\boldsymbol{y}_t$, is close to the equilibrium value.

Beginning with the first step, we note that $\max_{\boldsymbol{x}} \boldsymbol{x}^\top \boldsymbol{A}\overline{\boldsymbol{y}}_T \geq \max_{\boldsymbol{x}} \min_{\boldsymbol{y}} \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{y} =: A^*$ (where the last step follows via the minimax theorem), and so we get

$$\frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t^\top \boldsymbol{A}\boldsymbol{y}_t \geq A^* - \frac{R_T}{T}. \tag{25.4}$$

Similarly, we note that $\min_{\boldsymbol{y}} (\overline{\boldsymbol{x}}_T)^\top \boldsymbol{A}\boldsymbol{y} \leq \min_{\boldsymbol{y}} \max_{\boldsymbol{x}} \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{y} =: A^*$, where again the last step follows via the minimax theorem. This gives us

$$\frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t^\top \boldsymbol{A}\boldsymbol{y}_t \leq A^* + \frac{R_T}{T}. \tag{25.5}$$

Thus, Equations (25.4) and (25.5) respectively lower bound and upper bound the algorithm's performance by the value A^*. Putting these together gives us

$$\left| \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t^\top \boldsymbol{A}\boldsymbol{y}_t - A^* \right| \leq \frac{R_T}{T}. \tag{25.6}$$

Next, we show that the tuple $(\overline{\boldsymbol{x}}_T, \overline{\boldsymbol{y}}_T)$ achieves performance close to that of $\frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t^\top \boldsymbol{A}\boldsymbol{y}_t$. To do this, we note that $\min_{\boldsymbol{y}} (\overline{\boldsymbol{x}}_T)^\top \boldsymbol{A}\boldsymbol{y} \leq (\overline{\boldsymbol{x}}_T)^\top \boldsymbol{A}\overline{\boldsymbol{y}}_T$. Plugging this into Equation (25.3) then gives us

$$(\overline{\boldsymbol{x}}_T)^\top \boldsymbol{A}\overline{\boldsymbol{y}}_T \geq \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t^\top \boldsymbol{A}\boldsymbol{y}_t - \frac{R_T}{T}.$$

Similarly, we have that $\max_{\boldsymbol{x}} \boldsymbol{x}^\top \boldsymbol{A}\overline{\boldsymbol{y}}_T \geq (\overline{\boldsymbol{x}}_T)^\top \boldsymbol{A}\overline{\boldsymbol{y}}_T$, and plugging this into Equation (25.2) then gives us

$$(\overline{\boldsymbol{x}}_T)^\top \boldsymbol{A}\overline{\boldsymbol{y}}_T \leq \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t^\top \boldsymbol{A}\boldsymbol{y}_t + \frac{R_T}{T}.$$

Putting these together gives us

$$\left|(\overline{\boldsymbol{x}}_T)^\top \boldsymbol{A}\overline{\boldsymbol{y}}_T - \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t^\top \boldsymbol{A}\boldsymbol{y}_t\right| \leq \frac{R_T}{T}. \tag{25.7}$$

Finally, putting together Equations (25.6) and (25.7) via the triangle inequality gives us the requisite statement. ■

25.3.3 Implications for convex-concave min-max optimization

Theorem 5 implies a remarkably simple *algorithm* for computing the Nash equilibrium of a zero-sum game: simply allowing each player to play a no-regret algorithm, such as multiplicative weights or FTPL, and accumulate the time-average of the ensuing strategies! This algorithmic principle turns out to work well beyond a finite-game setting. In particular, consider the following *min-max optimization* problem

$$\min_{\boldsymbol{x}} \max_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y}) \tag{25.8}$$

that, as we saw in Example 2, arises in several contexts in modern machine learning. Our goal is to solve the above min-max optimization problem and find a point $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ that achieves the min-max value of Equation (25.8). For example, in adversarial training, $\boldsymbol{x}$ constitutes the parameters of a model and $\boldsymbol{y}$ constitutes a perturbation to input data.

For arbitrary functions $f(\cdot, \cdot)$, solving the objective in Equation (25.8) is significantly tougher than computing a Nash equilibrium in a finite zero-sum game (in fact, in the worst case it is known to be computationally hard)! However, it turns out that we can generalize the ideas in Section 25.3 to solve this objective if $f(\cdot, \cdot)$ has the following *convex-concave structure*:

- For any $\boldsymbol{y}$, $f(\cdot, \boldsymbol{y})$ is convex.
- For any $\boldsymbol{x}$, $f(\boldsymbol{x}, \cdot)$ is concave.

In this setting, a generalization of von-Neumann's minimax theorem (called Sion's minimax theorem) implies that $\min_{\boldsymbol{x}} \max_{\boldsymbol{y}} f(x, y) = \max_{\boldsymbol{y}} \min_{\boldsymbol{x}} f(x, y)$. Moreover, a generalization of the theory developed in Section 25.3 implies the following (informally stated):

*If the "min player" (who controls $\boldsymbol{x}$) uses an online convex optimization algorithm to minimize her losses (given by $\ell_t(\boldsymbol{x}) = f(\boldsymbol{x}, \boldsymbol{y}_t)$), and the "max player" (who controls $\boldsymbol{y}$) uses an online **concave** optimization algorithm to **maximize** his rewards (given by $g_t(\boldsymbol{y}) = f(\boldsymbol{x}_t, \boldsymbol{y})$), then their **averaged** iterates $\overline{\boldsymbol{x}}_T, \overline{\boldsymbol{y}}_T$ will approximately solve Equation (25.8). This is also commonly called an approximate Nash equilibrium of the convex-concave game.*

This result can be proved via a generalization of Theorem 5 with some extra steps (that independently use the convex-concave structure in f), and has some profound implications. For example, considering the case of adversarial training with simple ML models such as

linear models (which can be verified to yield convex-concave objectives), this tells us that adversarial training is no harder than regular training (without adversarial perturbations to the data), the only difference being that we need to *average our iterates*.

In fact, this seminal result forms the basis for several algorithms used for adversarial training (and also the more complex task of training of GANs) in practice; in particular, the *gradient-descent-ascent algorithm*. (This is in fact much faster in terms of iteration complexity, than solving the inner maximization for every possible value of $\boldsymbol{x}$, and minimizing the ensuing function $\phi(\boldsymbol{x}) := \max_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y})$.) Of course, in practice the associated min-max optimization problems are highly non-convex (and non-concave), and so the above theory does not apply at all. Training of GANs, in particular, is known to be particularly difficult and empirically susceptible to several issues. In general, tractable nonconvex-nonconcave min-max optimization is an open and highly important research problem.

25.4. Additional references and notes

- The roots of Theorem 5 are in the work of James Hannan in the 1950's Hannan (1957); in fact, Hannan invented the concept of no-regret as we study it today. The simple proof that we have provided is inspired by the approach taken in Freund and Schapire (1999).

- It is possible to achieve rates of convergence to Nash equilibrium that are even faster than $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$. In particular, an "optimistic" variant of multiplicative weights/online gradient descent that counts the last iterate twice (informally speaking) provably achieves a much faster rate of convergence to Nash equilibrium, given by $\mathcal{O}\left(\frac{1}{T}\right)$. This result was first shown in Rakhlin and Sridharan (2013), and related results are also provided by Daskalakis et al. (2011).

- While the *averaged iterates* $(\overline{\boldsymbol{x}}_T, \overline{\boldsymbol{y}}_T)$ are shown to converge to Nash as a very general consequence of the no-regret guarantee, the *day-to-day behavior* is a far more subtle issue. That is, does $\boldsymbol{x}_T, \boldsymbol{y}_T$ itself approach the Nash equilibrium? As we will briefly see next lecture, evidence for this is mixed, and only some no-regret algorithms can ensure that the day-to-day behavior itself approaches equilibrium behavior.

- Sion's minimax theorem implies *strong Lagrangian duality*, which is a cornerstone of efficiently solving constrained convex optimization problems.

References

Constantinos Daskalakis, Alan Deckelbaum, and Anthony Kim. Near-optimal no-regret algorithms for zero-sum games. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 235–254. SIAM, 2011.

Yoav Freund and Robert E Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.

James Hannan. Approximation to Bayes risk in repeated play. *Contributions to the Theory of Games*, 3:97–139, 1957.

Sasha Rakhlin and Karthik Sridharan. Optimization, learning, and games with predictable sequences. In *Advances in Neural Information Processing Systems 26*, pages 3066–3074, 2013.