Lecture 17

# *No-Regret Dynamics*

This lecture studies a second fundamental class of dynamics, *no-regret dynamics*. While best-response dynamics can only converge to a pure Nash equilibrium and is most relevant for potential games, no-regret dynamics converges to the set of coarse correlated equilibria in arbitrary finite games.

Section 17.1 considers a single decision maker playing a game online against an adversary, and defines no-regret algorithms.[1] Section 17.2 presents the multiplicative weights algorithm, and Section 17.3 proves that it is a no-regret algorithm. Section 17.4 defines no-regret dynamics in multi-agent games, and proves that it converges to the set of coarse correlated equilibria.

## 17.1 Online Decision Making

### 17.1.1 The Model

Consider a set $A$ of $n \geq 2$ actions and a time horizon $T \geq 1$, both known in advance to a decision maker. For example, $A$ could represent different investment strategies, or different driving routes between home and work. When we return to multi-agent games (Section 17.4), the action set will be the strategy set of a single agent, with the consequence of each action determined by the strategies chosen by all of the other agents.

We consider the following setup.[2]

[1] In this context, "online" means that the protagonist must make a sequence of decisions without knowledge of the future.

[2] For extensions to costs in an interval $[-c_{\max}, c_{\max}]$, see Exercise 17.1.

**Online Decision Making**

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

- a decision maker picks a probability distribution $p^t$ over her actions $A$
- an adversary picks a cost vector $c^t : A \to [-1, 1]$
- an action $a^t$ is chosen according to the distribution $p^t$, and the decision maker incurs cost $c^t(a^t)$
- the decision maker learns $c^t$, the entire cost vector[3]

An *online decision-making algorithm* specifies for each $t$ the probability distribution $p^t$, as a function of the cost vectors $c^1, \ldots, c^{t-1}$ and realized actions $a^1, \ldots, a^{t-1}$ of the first $t-1$ time steps. An *adversary* for such an algorithm $\mathcal{A}$ specifies for each $t$ the cost vector $c^t$, as a function of the probability distributions $p^1, \ldots, p^t$ used by $\mathcal{A}$ on the first $t$ days and the realized actions $a^1, \ldots, a^{t-1}$ of the first $t-1$ days. We evaluate the performance of an online decision-making algorithm by its expected cost (over the realized actions) with respect to a worst-case adversary. Negative costs are allowed, and can be used to model payoffs.

### 17.1.2 Definitions and Examples

We seek a "good" online decision-making algorithm. But the setup seems a bit unfair, no? The adversary is allowed to choose each cost function $c^t$ *after* the decision maker has committed to her probability distribution $p^t$. With such asymmetry, what kind of guarantee can we hope for? This section gives three examples that establish limitations on what is possible.

The first example shows that there is no hope of achieving cost close to that of the best action sequence in hindsight. This benchmark $\sum_{t=1}^{T} \min_{a \in A} c^t(a)$ is just too strong.

[3] The guarantees presented in this lecture carry over, with somewhat worse bounds and more complex algorithms, to the *bandit model* in which the decision maker only learns the cost of her chosen action.

**Example 17.1 (Comparing to the Best Action Sequence)** Suppose $A = \{1, 2\}$ and fix an arbitrary online decision-making algorithm. Each day $t$, the adversary chooses the cost vector $c^t$ as follows: if the algorithm chooses a distribution $p^t$ for which the probability on action 1 is at least $\frac{1}{2}$, then $c^t$ is set to the vector $(1, 0)$. Otherwise, the adversary sets $c^t$ equal to $(0, 1)$. This adversary forces the expected cost of the algorithm to be at least $\frac{T}{2}$ while ensuring that the cost of the best action sequence in hindsight is 0.

Example 17.1 motivates the following important definitions. Rather than comparing the expected cost of an algorithm to that of the best action *sequence* in hindsight, we compare it to the cost incurred by the best *fixed action* in hindsight. That is, we change our benchmark from $\sum_{t=1}^{T} \min_{a \in A} c^t(a)$ to $\min_{a \in A} \sum_{t=1}^{T} c^t(a)$.

**Definition 17.2 (Regret)** Fix cost vectors $c^1, \ldots, c^T$. The *regret* of the action sequence $a^1, \ldots, a^T$ is

$$\frac{1}{T}\left[\sum_{t=1}^{T} c^t(a^t) - \min_{a \in A} \sum_{t=1}^{T} c^t(a)\right]. \tag{17.1}$$

The quantity in (17.1) is sometimes called *external* regret.[4] Lecture 18 discusses swap regret, a more stringent notion.

**Definition 17.3 (No-Regret Algorithm)** An online decision-making algorithm $\mathcal{A}$ has *no regret* if for every $\epsilon > 0$ there exists a sufficiently large time horizon $T = T(\epsilon)$ such that, for every adversary for $\mathcal{A}$, in expectation over the action realizations, the regret (17.1) is at most $\epsilon$.

In Definition 17.3, we think of the number $n$ of actions as fixed, and the time horizon $T$ tending to infinity.[5]

This lecture adopts the no-regret guarantee of Definition 17.3 as the holy grail in the design of online decision-making algorithms. The

[4] This quantity can be negative, but it is positive for worst-case adversaries (Example 17.5).

[5] Strictly speaking, Definition 17.3 concerns a *family* of online decision-making algorithms, one for each value of $T$ (with the action set $A$ fixed). See Remark 17.8 and Exercise 17.2 for extensions to the scenario where $T$ is not known to the decision maker a priori.

first reason is that this goal can be achieved by simple and natural learning algorithms (Section 17.2). The second reason is that the goal is nontrivial: as the following examples make clear, some ingenuity is required to achieve it. The third reason is that, when we pass to multi-agent games in Section 17.4, the no-regret guarantee translates directly to the coarse correlated equilibrium conditions (Definition 13.5).

One natural online decision-making algorithm is *follow-the-leader (FTL)*, which at time step $t$ chooses the action $a$ with minimum cumulative cost $\sum_{u=1}^{t-1} c^u(a)$ so far. The next example shows that FTL is not a no-regret algorithm, and more generally rules out any deterministic no-regret algorithm.

**Example 17.4 (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 cost of action $a^t$ to 1, and the cost of every other action to 0. Then, the cost of the algorithm is $T$ while the cost of the best action in hindsight is at most $\frac{T}{n}$. Even when there are only 2 actions, for arbitrarily large $T$, the worst-case regret of the algorithm is at least $\frac{1}{2}$.

For randomized algorithms, the next example limits the rate at which regret can vanish as the time horizon $T$ grows.

**Example 17.5 ($\sqrt{(\ln n)/T}$ Lower Bound on Regret)** Suppose there are $n = 2$ actions, and that we choose each cost vector $c^t$ independently and equally likely to be $(1, 0)$ or $(0, 1)$. No matter how smart or dumb an online decision-making algorithm is, with respect to this random choice of cost vectors, its expected cost at each time step is exactly $\frac{1}{2}$ and its expected cumulative cost is $\frac{T}{2}$. The expected cumulative cost of the best fixed action in hindsight is only $\frac{T}{2} - b\sqrt{T}$, where $b$ is some constant independent of $T$. 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 $\frac{1}{2}\sqrt{T}$.

Fix an online decision-making algorithm $\mathcal{A}$. A random choice of cost vectors causes $\mathcal{A}$ to experience expected regret at least $b/\sqrt{T}$, where the expectation is over both the random choice of cost vectors and the action realizations. At least one choice of cost vectors induces

an adversary that causes $\mathcal{A}$ to have expected regret at least $b/\sqrt{T}$, where the expectation is over the action realizations.

A similar argument shows that, with $n$ actions, the expected regret of an online decision-making algorithm cannot vanish faster than $b\sqrt{(\ln n)/T}$, where $b > 0$ is some constant independent of $n$ and $T$ (Problem 17.1).

## 17.2 The Multiplicative Weights Algorithm

The most important result in this lecture is that *no-regret algorithms exist*. Lecture 18 shows that this fact alone has some amazing consequences. Even better, there are simple and natural such algorithms. While not a literal description of human behavior, the guiding principles behind such algorithms are recognizable from the way many people learn and make decisions. Finally, the algorithm discussed next has optimal worst-case expected regret, matching the lower bound in Example 17.5 up to constant factors.

**Theorem 17.6 (No-Regret Algorithms Exist)** *For every set $A$ of $n$ actions and time horizon $T \geq 4 \ln n$, there is an online decision-making algorithm that, for every adversary, has expected regret at most $2\sqrt{(\ln n)/T}$.*

An immediate corollary is that the number of time steps needed to drive the expected regret down to a small constant is only logarithmic in the number of actions.

**Corollary 17.7 (Logarithmic Number of Steps Suffice)** *For every $\epsilon \in (0, 1]$, set $A$ of $n$ actions and time horizon $T \geq (4 \ln n)/\epsilon^2$, there is an online decision-making algorithm that, for every adversary, has expected regret at most $\epsilon$.*

The guarantees of Theorem 17.6 and Corollary 17.7 are achieved in particular by the *multiplicative weights (MW)* algorithm.[6] Its design follows two guiding principles.

[6] Variants of this algorithm have been rediscovered many times; see the Notes.

**No-Regret Algorithm Design Principles**

1. Past performance of actions should guide which action is chosen at each time step, with the probability of choosing an action decreasing in its cumulative cost.

2. The probability of choosing a poorly performing action should decrease at an exponential rate.

The first principle is essential for obtaining a no-regret algorithm, and the second for optimal regret bounds.

The MW algorithm maintains a weight, intuitively a "credibility," for each action. At each time step the algorithm chooses an action with probability proportional to its current weight.

**Multiplicative Weights (MW) Algorithm**

initialize $w^1(a) = 1$ for every $a \in A$

**for** each time step $t = 1, 2, \ldots, T$ **do**

use the distribution $p^t = w^t/\Gamma^t$ over actions, where $\Gamma^t = \sum_{a \in A} w^t(a)$ is the sum of the weights

given the cost vector $c^t$, for every action $a \in A$ use the formula $w^{t+1}(a) = w^t(a) \cdot (1 - \eta c^t(a))$ to update its weight

For example, if all costs are either -1, 0, or 1, then the weight of each action $a$ either stays the same (if $c^t(a) = 0$) or gets multiplied by $1 - \eta$ (if $c^t(a) = 1$) or $1 + \eta$ (if $c^t(a) = -1$). The parameter $\eta$, which is sometimes called the "learning rate," lies between 0 and $\frac{1}{2}$, and is chosen at the end of the proof of Theorem 17.6 as a function of $n$ and $T$. When $\eta$ is close to 0, the distributions $p^t$ stay close to the uniform distribution. Thus small values of $\eta$ encourage exploration. As $\eta$ tends to 1, the distributions $p^t$ increasingly favor the actions with the smallest cumulative cost so far. Thus large values of $\eta$ encourage exploitation, and the parameter provides a knob for interpolating between these two extremes. The MW algorithm is simple to implement, as the only requirement is to maintain a weight for each action.

## *17.3 Proof of Theorem 17.6

### 17.3.1 Adaptive vs. Oblivious Adversaries

In the definition of an adversary for an online decision-making algorithm (Section 17.1), the cost vector $c^t$ can depend on what happened in the first $t-1$ time steps. Such adversaries are called *adaptive*. An *oblivious adversary* for an algorithm specifies the entire sequence $c^1, \ldots, c^T$ of cost vectors in advance, before any actions are realized.

To prove Theorem 17.6 for the MW algorithm, we only need to consider oblivious adversaries. The reason is that the behavior of the MW algorithm is independent of the realized actions, with each distribution $p^t$ chosen by the algorithm a deterministic function of $c^1, \ldots, c^{t-1}$. Thus, to maximize the expected regret of the MW algorithm, there is no reason for an adversary to condition its cost vectors on previously realized actions. Similarly, there is no need for an adversary for the MW algorithm to condition a cost vector $c^t$ explicitly on the distributions $p^1, \ldots, p^t$, since these distributions are uniquely determined by the adversary's previous cost vectors $c^1, \ldots, c^{t-1}$.

### 17.3.2 The Analysis

Fix a set $A$ of $n$ actions and a time horizon $T \geq 4 \ln n$. Fix an oblivious adversary, or equivalently a sequence $c^1, \ldots, c^T$ of cost vectors. This fixes the corresponding sequence $p^1, \ldots, p^T$ of probability distributions used by the MW algorithm. Recall that $\Gamma^t = \sum_{a \in A} w^t(a)$ denotes the sum of the actions' weights in the MW algorithm at the beginning of time step $t$. The proof plan is to relate the only two quantities that we care about, the expected cost of the MW algorithm and the cost of the best fixed action, to the intermediate quantity $\Gamma^{T+1}$.

The first step, and the step that is special to the MW algorithm, shows that the sum of the weights $\Gamma^t$ evolves together with the expected cost incurred by the algorithm. Letting $\nu^t$ denote the expected cost of the MW algorithm at time step $t$, we have

$$\nu^t = \sum_{a \in A} p^t(a) \cdot c^t(a) = \sum_{a \in A} \frac{w^t(a)}{\Gamma^t} \cdot c^t(a). \tag{17.2}$$

We want to upper bound the sum of the $\nu^t$'s.

To understand $\Gamma^{t+1}$ as a function of $\Gamma^t$ and the expected cost (17.2), we derive

$$\begin{aligned}\Gamma^{t+1} &= \sum_{a \in A} w^{t+1}(a) \\ &= \sum_{a \in A} w^t(a) \cdot (1 - \eta c^t(a)) \\ &= \Gamma^t(1 - \eta \nu^t). \end{aligned} \tag{17.3}$$

For convenience, we'll bound this quantity from above, using the fact that $1 + x \leq e^x$ for all real-valued $x$ (Figure 17.1). Then,

$$\Gamma^{t+1} \leq \Gamma^t \cdot e^{-\eta \nu^t}$$

for each $t$ and hence

$$\Gamma^{T+1} \leq \underbrace{\Gamma^1}_{=n} \prod_{t=1}^{T} e^{-\eta \nu^t} = n \cdot e^{-\eta \sum_{t=1}^{T} \nu^t}. \tag{17.4}$$

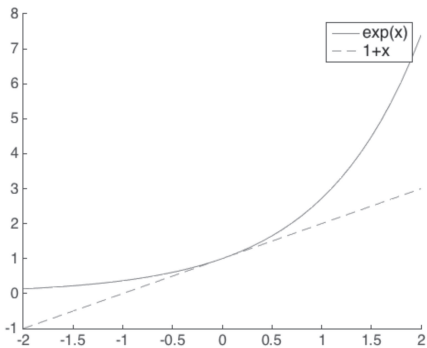


**Figure 17.1:** The inequality $1 + x \leq e^x$ holds for all real-valued $x$.

The second step is to show that if there is a good fixed action, then the weight of this action single-handedly shows that the final value $\Gamma^{T+1}$ is pretty big. This implies that the algorithm can only incur large cost if all fixed actions are bad.

Formally, let $OPT$ denote the cumulative cost $\sum_{t=1}^{T} c^t(a^*)$ of the best fixed action $a^*$ for the cost vector sequence. Then, since weights

are always nonnegative,

$$\begin{aligned}\Gamma^{T+1} &\geq w^{T+1}(a^*) \\ &= \underbrace{w^1(a^*)}_{=1}\prod_{t=1}^{T}(1-\eta c^t(a^*)). \end{aligned} \tag{17.5}$$

It is again convenient to approximate $1+x$ by an exponential function, this time from below. Figure 17.1 indicates that the two functions are close to each other for $x$ near 0. This can be made precise through the Taylor expansion

$$\ln(1-x) = -x - \tfrac{x^2}{2} - \tfrac{x^3}{3} - \tfrac{x^4}{4} - \cdots .$$

Provided $|x| \leq \frac{1}{2}$, we can obtain a lower bound of $-x-x^2$ on $\ln(1-x)$ by throwing out all terms of the expansion except the first two, and doubling the second term to compensate. Hence, $1-x \geq e^{-x-x^2}$ for $|x| \leq \frac{1}{2}$.

Since $\eta \leq \frac{1}{2}$ and $|c^t(a^*)| \leq 1$ for every $t$, we can combine this lower bound with (17.5) to obtain

$$\begin{aligned}\Gamma^{T+1} &\geq \prod_{t=1}^{T} e^{-\eta c^t(a^*)-\eta^2 c^t(a^*)^2} \\ &\geq e^{-\eta OPT-\eta^2 T}, \end{aligned} \tag{17.6}$$

where in (17.6) we're just using the crude estimate $c^t(a^*)^2 \leq 1$ for all $t$.

Through (17.4) and (17.6), we've connected the cumulative expected cost $\sum_{t=1}^{T}\nu^t$ of the MW algorithm with the cumulative cost $OPT$ of the best fixed action via the intermediate quantity $\Gamma^{T+1}$:

$$n \cdot e^{-\eta\sum_{t=1}^{T}\nu^t} \geq \Gamma^{T+1} \geq e^{-\eta OPT-\eta^2 T}.$$

Taking the natural logarithm of both sides and dividing through by $-\eta$ yields

$$\sum_{t=1}^{T}\nu^t \leq OPT + \eta T + \frac{\ln n}{\eta}. \tag{17.7}$$

Finally, we set the free parameter $\eta$. There are two error terms in (17.7), the first one corresponding to inaccurate learning (higher

for larger $\eta$), the second corresponding to learning overhead (higher for smaller $\eta$). To equalize the two terms, we choose $\eta = \sqrt{(\ln n)/T}$. As $T \geq 4 \ln n$, $\eta \leq \frac{1}{2}$, as required. The cumulative expected cost of the MW algorithm is then at most $2\sqrt{T \ln n}$ more than the cumulative cost of the best fixed action. This completes the proof of Theorem 17.6.

**Remark 17.8 (Unknown Time Horizons)** The choice of $\eta$ in the proof above assumes advance knowledge of the time horizon $T$. Minor modifications extend the multiplicative weights algorithm and its regret guarantee to the case where $T$ is not known a priori, with the "2" in Theorem 17.6 replaced by a modestly larger factor (Exercise 17.2).

## 17.4 No Regret and Coarse Correlated Equilibria

We now move from single-agent to multi-agent settings and study *no-regret dynamics* in finite games.

### 17.4.1 No-Regret Dynamics

We describe no-regret dynamics using the language of cost-minimization games (Section 13.1.1). There is an obvious analog for payoff-maximization games, with payoffs acting as negative costs.

**No-Regret Dynamics**

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

each agent $i$ independently chooses a mixed strategy $p_i^t$ using a no-regret algorithm, with actions corresponding to pure strategies

each agent $i$ receives a cost vector $c_i^t$, where $c_i^t(s_i)$ is the expected cost of the pure strategy $s_i$ given the mixed strategies chosen by the other agents:

$$c_i^t(s_i) = \mathbf{E}_{\mathbf{s}_{-i}^t \sim \sigma_{-i}^t}\left[C_i(s_i, \mathbf{s}_{-i}^t)\right],$$

where $\sigma_{-i}^t$ is the product distribution $\prod_{j \neq i} p_j^t$

For example, if every agent uses the MW algorithm, then in each iteration each agent simply updates the weight of each of her pure strategies. In this case, if every agent has at most $n$ strategies and costs lie in $[-c_{\max}, c_{\max}]$, then only $(4c_{\max}^2 \ln n)/\epsilon^2$ iterations of no-regret dynamics are required before every agent has expected regret at most $\epsilon$ (Theorem 17.6 and Exercise 17.1).

### 17.4.2 Convergence to Coarse Correlated Equilibria

The next result is simple but important: the time-averaged history of joint play under no-regret dynamics converges to the set of coarse correlated equilibria, the biggest set in our hierarchy of equilibrium concepts (Definition 13.5). This forges a fundamental connection between a static equilibrium concept and the outcomes generated by natural learning dynamics.

**Proposition 17.9 (No-Regret Dynamics Converges to CCE)** *Suppose that after $T$ iterations of no-regret dynamics, each agent $i = 1, 2, \ldots, k$ of a cost-minimization game has expected regret at most $\epsilon$. Let $\sigma^t = \prod_{i=1}^k p_i^t$ denote the outcome distribution at iteration $t$ and $\sigma = \frac{1}{T}\sum_{t=1}^T \sigma^t$ the time-averaged history of these distributions. Then $\sigma$ is an approximate coarse correlated equilibrium, in the sense that*

$$\mathbf{E}_{\mathbf{s}\sim\sigma}[C_i(\mathbf{s})] \leq \mathbf{E}_{\mathbf{s}\sim\sigma}\left[C_i(s_i', \mathbf{s}_{-i})\right] + \epsilon \tag{17.8}$$

*for every agent $i$ and unilateral deviation $s_i'$.*

*Proof:* By the definition of $\sigma$, for every agent $i$,

$$\mathbf{E}_{\mathbf{s}\sim\sigma}[C_i(\mathbf{s})] = \frac{1}{T}\sum_{t=1}^T \mathbf{E}_{\mathbf{s}\sim\sigma^t}[C_i(\mathbf{s})] \tag{17.9}$$

and

$$\mathbf{E}_{\mathbf{s}\sim\sigma}\left[C_i(s_i', \mathbf{s}_{-i})\right] = \frac{1}{T}\sum_{t=1}^T \mathbf{E}_{\mathbf{s}\sim\sigma^t}\left[C_i(s_i', \mathbf{s}_{-i})\right]. \tag{17.10}$$

The right-hand sides of (17.9) and (17.10) are the time-averaged expected costs of agent $i$ when playing according to her no-regret algorithm and when playing the fixed action $s_i'$ every iteration, respectively. Since every agent has regret at most $\epsilon$, the former is at most $\epsilon$

more than the latter. This verifies the approximate coarse correlated equilibrium conditions (17.8). ■

Proposition 17.9 gives a sense in which the coarse correlated equilibrium concept is particularly computationally tractable, and hence a relatively plausible prediction of agent behavior.

### 17.4.3 Final Remarks

The conventional interpretation of coarse correlated and correlated equilibria involves a third party who samples an outcome from the equilibrium distribution (Section 13.1). Proposition 17.9 demonstrates how such correlation arises endogenously when independent agents play the same game repeatedly. The correlation stems from the shared history of joint play.

The notion of approximate equilibrium in Proposition 17.9 concerns additive error, while Definitions 14.5 and 16.2 use relative error. These choices are primarily for technical convenience.

An alternative form of no-regret dynamics samples an outcome $\mathbf{s}^t$ according to the distribution $\sigma^t = \prod_{i=1}^k p_i^t$ at each iteration $t$, with agent $i$ receiving the cost vector $c_i^t$ with $c_i^t(s_i) = C_i(s_i, \mathbf{s}_{-i}^t)$ for each strategy $s_i \in S_i$. An analog of Proposition 17.9 holds for the uniform distribution $\sigma$ over the multi-set $\{\mathbf{s}^1, \ldots, \mathbf{s}^T\}$ of sampled outcomes, with the statement and the proof modified to accommodate sampling error. In these alternative dynamics, it is essential that agents use algorithms that have no regret with respect to adaptive adversaries (Section 17.3.1).

Lecture 14 shows that price-of-anarchy bounds for $(\lambda, \mu)$-smooth games (Definition 14.2) hold for all coarse correlated equilibria (Theorem 14.4) and degrade gracefully for approximate equilibria (Theorem 14.6). Thus, Proposition 17.9 suggests that such bounds should apply also to the time-averaged expected objective function value of an outcome sequence generated by no-regret dynamics. This is indeed the case (Exercise 17.3).

**Corollary 17.10 (POA Bounds for No-Regret Dynamics)** *Suppose that after $T$ iterations of no-regret dynamics, each of the $k$ agents of a $(\lambda, \mu)$-smooth cost-minimization game has expected regret at most $\epsilon$. If $\sigma^t = \prod_{i=1}^k p_i^t$ denotes the outcome distribution at*

*iteration $t$ and $\mathbf{s}^*$ an optimal outcome, then*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbf{E}_{\mathbf{s}\sim\sigma^t}[\text{cost}(\mathbf{s})] \leq \frac{\lambda}{1-\mu}\,\text{cost}(\mathbf{s}^*) + \frac{k\epsilon}{1-\mu}.$$

As $\epsilon \to 0$, this guarantee converges to $\frac{\lambda}{1-\mu}$, the standard price-of-anarchy bound for smooth games (Section 14.4).

**The Upshot**

- ☆ In each time step of an online decision-making problem, an algorithm chooses a probability distribution over actions and then an adversary reveals the cost of each action.
- ☆ The regret of an action sequence is the difference between the time-averaged costs of the sequence and of the best fixed action in hindsight.
- ☆ A no-regret algorithm guarantees expected regret tending to 0 as the time horizon tends to infinity.
- ☆ The multiplicative weights algorithm is a simple no-regret algorithm with optimal worst-case expected regret.
- ☆ In every iteration of no-regret dynamics, each agent independently chooses a mixed strategy using a no-regret algorithm.
- ☆ The time-averaged history of joint play in no-regret dynamics converges to the set of coarse correlated equilibria.
- ☆ Price-of-anarchy bounds in smooth games apply to the time-averaged expected objective function value of an outcome sequence generated by no-regret dynamics.

## Notes

The versions of the multiplicative weights algorithm and Theorem 17.6 described here are from Cesa-Bianchi et al. (2007). Many variants and extensions, including to the bandit model where the decision maker only learns the cost of the chosen action at each time step, are discussed by Cesa-Bianchi and Lugosi (2006) and Blum and Mansour (2007b). These sources, together with Foster and Vohra (1999) and Arora et al. (2012), also cover the history of online decision-making problems, no-regret algorithms, and important precursors to the multiplicative weights algorithm such as "randomized weighted majority" and "hedge." Key references include Blackwell (1956), Hannan (1957), Littlestone and Warmuth (1994), and Freund and Schapire (1997). Proposition 17.9 is already implicit in Hannan (1957). Problems 17.2 and 17.4 are from Littlestone (1988) and Kalai and Vempala (2005), respectively.

## Exercises

**Exercise 17.1** *(H)* Extend Corollary 17.7 to online decision-making problems where actions' costs lie in $[-c_{\max}, c_{\max}]$ rather than $[-1, 1]$, losing a factor of $c_{\max}^2$ in the number of time steps. You can assume that the value $c_{\max}$ is known in advance.

**Exercise 17.2** *(H)* The multiplicative weights algorithm requires advance knowledge of the time horizon $T$ to set the parameter $\eta$. Modify the algorithm so that it does not need to know $T$ a priori. Your algorithm should have expected regret at most $b\sqrt{(\ln n)/T}$ for all sufficiently large $T$ and for every adversary, where $b > 0$ is a constant independent of $n$ and $T$.

**Exercise 17.3** *(H)* Prove Corollary 17.10.

**Exercise 17.4** Proposition 17.9 proves that the time-averaged joint distribution $\frac{1}{T}\sum_{t=1}^{T} \sigma^t$ generated by no-regret dynamics is an approximate coarse correlated equilibrium, but it says nothing about the outcome distribution $\sigma^t$ in a given iteration $t$. Prove that such a distribution $\sigma^t$ is an approximate coarse correlated equilibrium if and

only if it is an approximate Nash equilibrium (with the same additive error term).

## Problems

**Problem 17.1** Consider an online decision-making problem with $n$ actions. Prove that the worst-case expected regret of an online decision-making algorithm cannot vanish faster than $b\sqrt{(\ln n)/T}$, where $b > 0$ is some constant independent of $n$ and $T$.

**Problem 17.2** This problem considers a variant of the online decision-making problem. There are $n$ "experts," where $n$ is a power of 2.

**Combining Expert Advice**

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

- each expert offers a prediction of the realization of a binary event (e.g., whether a stock will go up or down)
- a decision maker picks a probability distribution $p^t$ over the possible realizations 0 and 1 of the event
- the actual realization $r^t \in \{0, 1\}$ of the event is revealed
- a 0 or 1 is chosen according to the distribution $p^t$, and a *mistake* occurs whenever it is different from $r^t$

You are promised that there is at least one omniscient expert who makes a correct prediction at every time step.

(a) *(H)* A deterministic algorithm always assigns all of the probability mass in $p^t$ to one of 0 or 1. Prove that the minimum worst-case number of mistakes that a deterministic algorithm can make is precisely $\log_2 n$.

(b) Prove that for every randomized algorithm, there is a sequence of expert predictions and event realizations such that the expected number of mistakes made by the algorithm is at least $\frac{1}{2}\log_2 n$.

(c) *(H)* Prove that there is a randomized algorithm such that, for every sequence of expert predictions and event realizations, the expected number of mistakes is at most $b \log_2 n$, where $b < 1$ is a constant independent of $n$. How small can you take $b$?

**Problem 17.3** *(H)* Consider a $k$-agent cost-minimization game in which no agent $i$ incurs equal cost $C_i(\mathbf{s})$ in two different outcomes. Prove the following converse to Proposition 17.9: for every coarse correlated equilibrium $\sigma$ of the game, there exist choices of no-regret algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_k$ for the agents so that the time-averaged history of the corresponding no-regret dynamics converges to $\sigma$ as the number of iterations $T$ tends to infinity.

**Problem 17.4** Example 17.4 shows that the follow-the-leader (FTL) algorithm, and more generally every deterministic algorithm, fails to have no regret. This problem outlines a randomized variant of FTL, the *follow-the-perturbed-leader (FTPL)* algorithm, with worst-case expected regret comparable to that of the multiplicative weights algorithm. We define each probability distribution $p^t$ over actions implicitly through a randomized subroutine.

**Follow-the-Perturbed-Leader (FTPL) Algorithm**

**for** each action $a \in A$ **do**
  independently sample a geometric random variable with parameter $\eta$,[7] denoted by $X_a$
**for** each time step $t = 1, 2, \ldots, T$ **do**
  choose the action $a$ that minimizes the perturbed cumulative cost $-2X_a + \sum_{u=1}^{t-1} c^u(a)$ so far

[7] Equivalently, when repeatedly flipping a coin that comes up "heads" with probability $\eta$, count the number of flips up to and including the first "heads."

Fix an oblivious adversary, meaning a sequence $c^1, \ldots, c^T$ of cost vectors. For convenience, assume that, at every time step $t$, there is no pair of actions whose (unperturbed) cumulative costs-so-far differ by an integer.

(a) *(H)* Prove that, at each time step $t = 1, 2, \ldots, T$, with probability at least $1 - \eta$, the smallest perturbed cumulative cost of an action prior to $t$ is more than 2 less than the second-smallest such perturbed cost.

(b) *(H)* As a thought experiment, consider the (unimplementable) algorithm that, at each time step $t$, picks the action that minimizes the perturbed cumulative cost $-2X_a + \sum_{u=1}^{t} c^u(a)$, *taking into account the current cost vector*. Prove that the regret of this algorithm is at most $\max_{a \in A} X_a$.

(c) Prove that $\mathbf{E}[\max_{a \in A} X_a] \leq b\eta^{-1} \ln n$, where $n$ is the number of actions and $b > 0$ is a constant independent of $\eta$ and $n$.

(d) *(H)* Prove that, for a suitable choice of $\eta$, the worst-case expected regret of the FTPL algorithm is at most $b\sqrt{(\ln n)/T}$, where $b > 0$ is a constant independent of $n$ and $T$.

(e) *(H)* How would you modify the FTPL algorithm and its analysis to achieve the same regret guarantee with respect to adaptive adversaries?