Aaron Roth

Learning in Games (and Games in Learning)

INCOMPLETE WORKING DRAFT

Contents

1

Basics of Sequential Decision Making

CONTENTS

The foundational algorithmic ideas underlying this course allow a sequential decision maker to make choices such that their performance, in hindsight, can be related to (and shown to be superior to) benchmark policies of various sorts. This turns out to be closely related to optimal play in zero sum games, and in particular, the fundamental "minimax theorem", which we will both prove using sequential decision making techniques, and use to derive sequential decision making techniques. In this chapter, we'll derive some of these basic algorithms from first principles.

1.1 Basic Definitions

The simplest setting we will consider involves a sequential decision maker who must choose amongst a set of k actions $\mathcal{A} = \{1, \ldots, k\}$ each day t. After making their choice, the decision maker learns a *cost* $c_i^t \in [0, 1]$ for each action i, and experiences the cost of the action that they chose. Their goal will be to select actions that result in small cumulative cost. This setting is often called the "Learning from Expert Advice", which is terminology that comes from imagining that the actions correspond to "experts" whose advice the algorithm must choose between. Formally:

Definition 1 (Learning from Expert Advice) *The learner has an action set $\mathcal{A} = \{1, \ldots, k\}$. In rounds $t = 1, \ldots T$:*

1. With knowledge of $c^1, \ldots, c^{t-1}$, The Learner *chooses a distribution over actions $p^t \in \Delta\mathcal{A}$.*

2. With knowledge of $p^1, \ldots, p^t$, the Adversary *chooses a vector of costs $c^t \in [0,1]^k$.*

3. The Learner experiences (expected) cost $c_L^t = \mathbb{E}_{i \sim p_t}[c_i^t]$

We let $\pi^t = \{(p^s, c^s)\}_{s=1}^t$ denote the record of interaction after t rounds. After T rounds of interaction, the cumulative (expected) cost of the learner is $C_L^T = \sum_{t=1}^T c_L^t$. The cumulative cost of each action i is $C_i^T = \sum_{t=1}^T c_i^t$.

Observe that in this setting there is an *adversary*, who may choose costs in arbitrary ways, rather than a distribution over costs that the algorithm could have some hope of learning. As a result, it is not immediately clear how we should evaluate algorithms for the learner in this scenario — there is no fixed "optimal action" or "optimal policy" to learn. Instead, we will compare the accumulated cost of the learner to the accumulated cost that various benchmark policies would have obtained for the same sequence of chosen costs. To the extent that the algorithm has higher cost than one of the benchmark policies, we will say that the algorithm *regrets* not having played that benchmark; our goal will be to design algorithms that do not have regret to any benchmark in some fixed class. The simplest class of benchmark policies is the set of *constant* policies, that always play the same fixed action.

Definition 2 (Regret to the Best Fixed Action) *The learner's regret to action $i \in \mathcal{A}$ after T rounds is:*

$$Reg(\pi^T, i) = C_L^T - C_i^T = \sum_{t=1}^T (c_L^t - c_i^t)$$

We say that the learner has regret to the best fixed action bounded by α if:

$$\max_{i \in \mathcal{A}} Reg(\pi^T, i) \leq \alpha$$

1.2 Warmup: The Halving Algorithm

To convince ourselves that obtaining diminishing regret to the best action in hindsight is possible (and maybe even easy!) we'll start by working out a special case in which the solution is especially simple: the case in which there is a single action i^* (unbeknownst to the learner) that is guaranteed to have 0 cost at every round: $c_{i^*}^t = 0$ for all t. In this case, all the learner has to do is identify which action is the hidden "perfect" action. The solution (a variant of what is called the Halving algorithm) is to always play at round

t the uniform distribution p^t over actions that have never yet had positive cost, up through round t. The key to the analysis is to relate the cost of the algorithm at each round to the decrease in the number of actions that have never yet been observed to have positive cost, so that intuitively, at any round on which we experience significant cost, we also make significant progress towards identifying the perfect action. In this way, we set up a win-win situation for ourselves: we must *either* make progress towards learning or else experience low cost.

Algorithm 1 The Halving Algorithm

For each action $i \in \mathcal{A}$, set $w_i^1 = 1$. Let $W^1 = \sum_i w_i^1$.
for $t = 1$ to T **do**
Play the distribution p^t defined as:

$$p_i^t = \frac{w_i^t}{W^t}$$

Observe costs c^t and update weights such that for each $i \in \mathcal{A}$:

$$w_i^{t+1} = w_i^t(1 - \mathbb{1}[c_i^t > 0]) \qquad W^{t+1} = \sum_i w_i^{t+1}$$

Theorem 1 *For any sequence of costs such that there exists a perfect action i^* such that $c_{i^*}^t = 0$ for all t, the Halving Algorithm has regret to the best fixed action that is at most:*

$$\max_{i \in A} Reg(\pi^T, i) = Reg(\pi^T, i^*) \leq \ln(k)$$

Proof 1 *In this special case, we have that $C_{i^*}^T = 0$, and so we must show that $C_L^T \leq \log(k)$. At round t, let $C^t = \{i \in \mathcal{A} : c_i^t > 0\}$ be the set of actions that have positive cost at round t, and let $W(C^t) = \sum_{i \in C^t} w_i^t$ be their cumulative weight. Since costs are upper bounded by 1, we have that the loss of our algorithm at round t is at most:*

$$c_L^t \leq \frac{W(C^t)}{W^t}$$

We also have that:

$$W^{t+1} = W^t - W(C^t)$$

and so in particular we can write:

$$c_L^t \leq \frac{W^t - W^{t+1}}{W^t} = 1 - \frac{W^{t+1}}{W^t}$$

We know that $W^1 = k$ and $W^{T+1} \geq 1$ (since there is some expert that never

experiences positive cost). So we have:

$$\begin{aligned} 1 &\leq W^{T+1} \\ &= k \cdot \prod_{t=1}^{T} \frac{W^{t+1}}{W^t} \\ &\leq k \prod_{t=1}^{T} (1 - c_L^t) \\ &\leq k \prod_{t=1}^{T} \exp(-c_L^t) \\ &= k \exp\left(- \sum_{t=1}^{T} c_L^t \right) \\ &= k \exp(-C_L^T) \end{aligned}$$

Solving for C_L^T, we find:

$$C_L^T \leq \ln(k)$$

1.3 The General Case: Multiplicative Weights

The Halving algorithm gets a terrific regret bound (that doesn't grow at all with T), but makes important use of the fact that we have assumed that there is one perfect action. In particular, it zeros out the weight of any action that has ever been observed to have positive cost. If there is no perfect action, then we will quickly end up in a situation in which the Halving algorithm has zeroed out the weight of all experts, and it will no longer have a well defined distribution to play!

The solution to this problem is simple: we will simply be more restrained when reducing the weight of actions that have high cost. We will reduce the weight of each action in proportion to its cost at each round, but will make sure never to zero out the weight of any action, in case past performance turns out not to be predictive of future performance. The new algorithm works in nearly the same way, but has a parameter η that controls how aggressively we downweight high cost actions.

Algorithm 2 The Multiplicative Weights Algorithm

For each action $i \in \mathcal{A}$, set $w_i^1 = 1$. Let $W^1 = \sum_i w_i^1$.
for $t = 1$ to T **do**
Play the distribution p^t defined as:

$$p_i^t = \frac{w_i^t}{W^t}$$

Observe costs c^t and update weights such that for each $i \in \mathcal{A}$:

$$w_i^{t+1} = w_i^t(1 - \eta c_i^t) \qquad W^{t+1} = \sum_i w_i^{t+1}$$

Theorem 2 *Fix $\eta \leq \frac{1}{2}$. For every sequence of costs and for every action $i \in \mathcal{A}$, the multiplicative weights algorithm obtains regret to action i:*

$$Reg(\pi^T, i) = C_L^T - C_i^T \leq \frac{\ln k}{\eta} + \eta C_i^T$$

Remark 1.3.1 *If we know that $\max_i C_i^T \leq B$, we can set $\eta = \sqrt{\frac{\ln k}{B}}$ and Theorem 2 gives us a regret bound of:*

$$\max_{i \in A} Reg(\pi^T, i) \leq 2\sqrt{B \cdot \ln(k)}$$

When $B = 4 \ln k$, this recovers (up to a factor of 4) the bound we proved from the Halving algorithm under less restrictive assumptions. Since costs $c_i^t \in [0, 1]$, we always *know that $\max_i C_i^T \leq T$, and so we have a worst-case regret bound of:*

$$\max_{i \in A} Reg(\pi^T, i) \leq 2\sqrt{T \cdot \ln(k)}$$

for every sequence of costs, without assumptions. This bound grows sublinearly with T, which means that the algorithm promises that its average *per-round cost approaches that of the best action in hindsight at a rate of $O\left(\sqrt{\frac{\ln k}{T}}\right)$.*

Proof 2 *We want to replicate the argument we used to analyze the halving algorithm. The key once more will be to relate the cost c_L^t that the learner experiences at each round to the decrease in weight. From the weight update rule, we have that:*

$$W^{t+1} = W^t - \eta \sum_i w_i^t c_i^t = W^t(1 - \eta c_L^t)$$

The last equality in this chain follows from the fact that:

$$c_L^t = \sum_i p_i^t c_i^t = \sum_i \frac{w_i^t}{W^t} c_i^t = \frac{1}{W^t} \sum_i w_i^t c_i^t$$

This gives us a way to express how much the cumulative weight W^t decreases at each round. We still have that $W^1 = k$. When we analyzed the halving algorithm, we also knew that $W^{T+1} \geq 1$, since there was at least one action that never had its weight reduced. That is no longer true — but we can still lower bound W^{T+1} in terms of the final weight w_i^{T+1} of any action i, which we can itself bound by the cumulative cost C_i^{T+1} of action i:

$$W^{T+1} \geq w_i^{T+1} = \prod_{t=1}^{T}(1 - \eta c_i^t) \geq \prod_{t=1}^{T}(1-\eta)^{c_i^t} = (1-\eta)^{C_i^T}$$

Here we used the fact that for $c \in [0,1]$, $(1-\eta)^c \leq (1-\eta c)$. Now mirroring our analysis of the Halving algorithm, we can calculate:

$$\begin{aligned}
(1-\eta)^{C_i^T} &\leq W^{T+1} \\
&= k\prod_{t=1}^{T}(1-\eta c_L^t) \\
&\leq k\prod_{t=1}^{T}\exp(-\eta c_L^t) \\
&= k\exp\left(-\eta\sum_{t=1}^{T} c_L^t\right) \\
&= k\exp(-\eta C_L^T)
\end{aligned}$$

Taking the log of both sides and solving for C_L^T we have that:

$$C_L^T \leq \frac{1}{\eta}\left(\ln(k) + C_i^T \ln\left(\frac{1}{1-\eta}\right)\right)$$

For $\eta \leq \frac{1}{2}$, $\ln\left(\frac{1}{1-\eta}\right) \leq \eta + \eta^2$. Therefore we have:

$$C_L^T \leq \frac{\ln k}{\eta} + (1+\eta)C_i^T$$

and so:

$$Reg(\pi^T, i) = C_L^T - C_i^T \leq \frac{\ln k}{\eta} + \eta C_i^T$$

Remark 1.3.2 (Rescaling the Costs) *We have analyzed Multiplicative Weights for costs $c_i^t \in [0,1]$. What if the costs fall into a different range? What if they can be negative? We make two observations: First, consistently translating the cost vector by adding a fixed constant to each coordinate does not change the regret of a sequence of actions: so if costs are in the range $[-C_1, C_2]$, we can always translate them so that they fall into the range $[0, C]$ for $C = C_1 + C_2$. Next, we can rescale costs by dividing them all by C, which*

scales them back into the range $[0, 1]$*, at which point we can apply the Multiplicative Weights Theorem 2. Of course, scaling the costs down by* C *correspondingly scales the regret down by* C*, and so the final regret bound will be* C *times larger than that stated in Theorem 2. The result is that we can guarantee a regret bound of* $2(C_1 + C_2)\sqrt{T \cdot \ln(k)}$ *when the costs lie in the range* $[-C_1, C_2]$ *for any* $C_1, C_2 \in \mathbb{R}^{\geq 0}$*.*

1.4 Large Action Spaces: Online Linear Optimization

The multiplicative weights algorithm lets us get diminishing regret to the best fixed action in some comparison class $\mathcal{A}$: the regret guarantee grows only logarithmically with $k = |\mathcal{A}|$, and so in principle, we get very strong guarantees even for enormous action spaces. However, the running time of the algorithm grows linearly with k, and so we might struggle to run the algorithm if $\mathcal{A}$ is truly enormous. What can we do in settings in which $\mathcal{A}$ is exponentially large in some natural dimension of our problem, or even continuously large?

A useful running example to keep in mind is the online shortest paths problem, which is defined on a graph $G = (V, E)$, with two distinguished vertices, the source s and the sink t. The action space for the learner corresponds to the set of all $s \to t$ paths P, which can be exponentially large in the number of vertices d in the graph. Every round, a cost (congestion) c_e is realized for each edge $e \in E$ in the graph, and the cost of a path is the sum of the costs of the edges it contains: $c_P = \sum_{e \in P} c_e$. The goal is to select paths so that the learner's regret to the best fixed action (path) is tending to 0. We could run the multiplicative weights algorithm with one action per path to solve this problem with $O(\sqrt{dT})$ regret bounds, but with running time exponential in d. Can we get similar guarantees with running time polynomial in d? We'll show how to solve this problem as a special case of the more general online linear optimization problem.

Definition 3 (Online Linear Optimization) *In the online linear optimization problem:*

1. *The Learner has an action space* $\mathcal{A} \subseteq \mathbb{R}^d$*, and*
2. *The Adversary has an action space* $\mathcal{C} \subseteq \mathbb{R}^d$*.*
3. *At each round* t*, the learner chooses an action* $a^t \in \Delta\mathcal{A}$ *and the adversary chooses an action* $c^t \in \mathcal{C}$*. The learner experiences cost* $c_L^t = \langle a^t, c^t \rangle$

Remark 1.4.1 *The online linear optimization setting generalizes the no-regret setting we have already studied, in which there are* d *actions,* $\mathcal{A}$ *corresponds to the set of standard basis vectors (each indicating playing one of*

the d actions), and $\Delta\mathcal{A}$ is the set of probability distributions over the individual actions.

It also allows us to represent the online shortest path problem: Here the dimension $d = |E|$, with one coordinate for each edge $e \in E$. The set of actions $\mathcal{A}$ corresponds to the set of $s \to t$ paths in the graph, with each path $P \in \mathcal{A}$ represented as its indicator vector, in which $P_e = 1$ for each edge e in the path and $P_e = 0$ otherwise. $\Delta\mathcal{A}$ here represents the set of probability distributions over paths.

In general, $\Delta\mathcal{A}$ represents the convex hull of $\mathcal{A}$, which corresponds to the set of probability distributions over elements of $\mathcal{A}$ if $\mathcal{A}$ is not already a convex set. If $\mathcal{A}$ is already a convex set then $\Delta\mathcal{A} = \mathcal{A}$, and the learner can choose an element of $\mathcal{A}$ at every round.

Since the action space $\mathcal{A}$ can have arbitrary, in general finding the best action even for a fixed cost vector c (a strictly easier problem) might be computationally hard. Thus we will assume that we have an oracle to solve static linear optimization problems over $\mathcal{A}$.

Definition 4 *A (static) linear optimization oracle for $\mathcal{A}$ allows us to solve the following optimization problem for any vector $c \in \mathbb{R}$:*

$$a^*(c) \in \arg\min_{a \in \mathcal{A}} \langle a, c \rangle$$

How can we make use of a static linear optimization oracle to solve the online linear optimization problem? A natural first attempt is to simply optimize at every round t to play the action $a_t \in \mathcal{A}$ that has done best so far. We might call this approach "Follow the Leader".

Algorithm 3 Follow the Leader

for $t = 1$ to T **do**

Let:

$$c^{<t} = \sum_{s=1}^{t-1} c^s$$

Be the cumulative cost observed so far.

Select the action:

$$a^t = \arg\min_{a \in \mathcal{A}} \langle a, c^{<t} \rangle$$

Although this is a natural algorithm, it can behave quite poorly. Here is an example, in which $\mathcal{A} = \{(1,0),(0,1)\}$, and the sequence of costs is:

$$c^1 = (1/2, 0) \;\; c^2 = (0,1) \;\; c^3 = (1,0) \;\; c^4 = (0,1) \;\; c^5 = (1,0), \;\; c^6 = (0,1), \ldots$$

Follow the leader chooses the sequence of actions

$$a^1 = (1,0), \;\; a^2 = (0,1) \;\; a^3 = (1,0), \;\; a^4 = (0,1), \;\; a^5 = (1,0), \;\; a^6 = (0,1), \ldots$$

which is the worst possible sequence of actions, and incurs cost $T - 1/2$ after T rounds. On the other hand, the best fixed action in hindsight accumulates cost only $T/2$. So we need a new idea if we want to be able to bound our regret to the best fixed action by something that grows sublinearly with T.

The fix turns out to be simple, however: essentially the only thing we need to do is avoid the oscilattory behavior of the bad example we just saw. The way to do this is to add random noise to the cumulative observed costs before invoking the linear optimization oracle to select the next action. The final algorithm is called follow the *perturbed* leader.

Algorithm 4 Follow the Perturbed Leader

for $t = 1$ to T **do**

Let:

$$c^{<t} = \sum_{s=1}^{t-1} c^s$$

Be the cumulative cost observed so far.

Let $N^t \sim U[0, 1/\epsilon]^d$ be a uniformly random noise vector.

Select the action:

$$a^t = \arg\min_{a \in \mathcal{A}} \langle a, c^{<t} + N^t \rangle$$

Theorem 3 *The expected regret of Follow the Perturbed Leader to each action $a \in \mathcal{A}$ can be bounded as:*

$$\sum_{t=1}^{T} \left(\mathbb{E}[\langle a^t, c^t \rangle] - \langle a, c^t \rangle \right) \leq \frac{A}{\epsilon} + 2CRT\epsilon$$

where: $A = \max_{a,a' \in \mathcal{A}} ||a - a'||_1$, $C = \max_{c \in \mathcal{C}} ||c||_1$ and $R = \max_{a \in \mathcal{A}, c \in \mathcal{C}} |\langle a, c \rangle|$

Remark 1.4.2 *If we set $\epsilon = \sqrt{\frac{A}{2CRT}}$ then we get that Follow the Perturbed Leader has a regret bound of:*

$$\min_{a \in \mathcal{A}} \sum_{t=1}^{T} \left(\mathbb{E}[\langle a^t, c^t \rangle] - \langle a, c^t \rangle \right) \leq \sqrt{8CART}$$

Proof 3 (Proof of Theorem 3) *To analyze follow the perturbed leader, we will analyze a sequence of hypothetical algorithms, ending with follow the perturbed leader, and relate their regret bounds to each other.*

The first algorithm we will consider is called "be the leader", and selects action $\hat{a}^t$ at each round where:

$$\hat{a}^t = \arg\min_{a \in \mathcal{A}} \langle a, c^{<t+1} \rangle$$

Note that "be the leader" selects its action by optimizing for the cumulative loss up through and including *round t: $c^{<t+1}$. In other words, it is playing "follow the leader", but shifted one step forward in time. So it is not an implementable algorithm (because it requires seeing one step into the future), but we can still analyze its behavior.*

Lemma 1.4.1 *Be the leader has non-positive regret to every action $a \in \mathcal{A}$:*

$$\sum_{t=1}^{T} \langle \hat{a}^t, c^t \rangle \leq \sum_{t=1}^{T} \langle a, c^t \rangle$$

Proof 4 *We prove this by induction on T. The base case of $T = 1$ follows from the definition of $\hat{a}_1 = \arg\min_{a \in \mathcal{A}} \langle a, c^1 \rangle$. We now assume that the claim holds for $T = k$, and show that it holds for $T = k + 1$ as well:*

$$\begin{aligned}
\sum_{t=1}^{k+1} \langle \hat{a}^t, c^t \rangle &= \sum_{t=1}^{k} \langle \hat{a}^t, c^t \rangle + \langle \hat{a}^{k+1}, c^{k+1} \rangle \\
&\leq \sum_{t=1}^{k} \langle \hat{a}^{k+1}, c^t \rangle + \langle \hat{a}^{k+1}, c^{k+1} \rangle \\
&= \langle \hat{a}^{k+1}, \sum_{t=1}^{k+1} c^t \rangle \\
&\leq \langle a, \sum_{t=1}^{k+1} c^t \rangle
\end{aligned}$$

Here the first inequality follows from the induction hypothesis, and the last inequality follows from the fact that $\hat{a}^{k+1}$ is selected to be the minimizer of $\langle a, \sum_{t=1}^{k+1} c^t \rangle = \langle a, c^{<k+1} \rangle$

The next algorithm we will consider on our journey towards Follow the Perturbed Leader is called "Be the Perturbed Leader" and selects action $\tilde{a}^t$ at each round where:

$$\tilde{a}^t = \arg\min_{a \in \mathcal{A}} \langle a, c^{<t+1} + N^t \rangle$$

In other words, the algorithm uses the same uniform perturbations as Follow the Perturbed Leader, but perturbs the one-step-lookahead cumulative costs of Be the Leader. The next step of the argument is to show that perturbations don't hurt the performance of Be the Perturbed Leader too much:

Lemma 1.4.2 *For every sequence of cost vectors, Be the Perturbed Leader has expected regret to every action $a \in \mathcal{A}$ bounded as:*

$$\mathop{\mathbb{E}}_{N} \left[\sum_{t=1}^{T} \langle \tilde{a}^t, c^t \rangle \right] - \sum_{t=1}^{T} \langle a, c^t \rangle \leq \frac{A}{\epsilon}$$

Where $A = \max_{a, a' \in \mathcal{A}} ||a - a'||_1$

Proof 5 *Since we are bounding the expected cost of Be the Perturbed Leader, we can imagine that the perturbations are coupled such that* $N^1 = N^2 = \ldots = N^T = N$ *(with each still marginally distributed as* $N \sim U[0, 1/\epsilon]^d$*) — this does not effect the expected cost of the algorithm. With this observation, we can view "Be the Perturbed Leader" as actually playing "Be the Leader" in which there is an imagined "round 0" with costs* $c^0 = N$ *— this is equivalent, since the chosen action:*

$$\tilde{a}^t = \arg\min_{a \in \mathcal{A}} \langle a, \sum_{s=1}^{t} c^t + N^t \rangle = \arg\min_{a \in \mathcal{A}} \langle a, \sum_{s=0}^{t} c^t \rangle$$

Thus we can apply the guarantee of Be the Leader from Lemma 1.4.1 to conclude that:

$$\sum_{t=0}^{T} \langle \tilde{a}^t, c^t \rangle - \sum_{t=0}^{T} \langle a, c^t \rangle \leq 0$$

To translate this into a regret bound for Be the Perturbed Leader, we need to isolate the imagined "round 0" terms:

$$\sum_{t=1}^{T} \langle \tilde{a}^t, c^t \rangle - \sum_{t=1}^{T} \langle a, c^t \rangle \leq \langle a, c^0 \rangle - \langle \tilde{a}^0, c^0 \rangle = \langle a - \tilde{a}^0, c^0 \rangle \leq \frac{A}{\epsilon}$$

Here the last inequality follows from the fact that for any $a, a' \in \mathcal{A}$*:*

$$\langle a - a', c^0 \rangle \leq ||a - a'||_1 \cdot ||c^0||_\infty \leq A \cdot \frac{1}{\epsilon}$$

since c^0 *is a vector with coordinates bounded by* $1/\epsilon$*.*

All that remains is to relate the expected regret of Be the Perturbed Leader with that of Follow the Perturbed Leader. Be the Perturbed Leader and Follow the Perturbed Leader choose actions by optimizing for a cumulative loss vector that differ only in a single round (Be the Perturbed Leader aggregates over one additional round); the idea is that the noise added drowns out this difference, and so the two must have similar regret. Since we know that the regret of Be the Perturbed Leader is small, so must be the loss of Follow the Perturbed Leader.

$N^t, \tilde{N}^t \sim U[0, 1/\epsilon]$*. Fix a round* t*, and let* $p^t = c^{<t} + N^t$ *be the noisy accumulated vector of costs that Follow the Perturbed Leader optimizes for, and let* $\tilde{p}^t = c^{<t+1} + \tilde{N}^t$ *be the noisy accumulated vector of costs that Be the Perturbed Leader optimizes for. Observe that* p^t *is uniformly distributed in* $c^{<t} + [0, 1/\epsilon]^d$ *and that* $\tilde{p}^t$ *is uniformly distributed in* $c^{<t+1} + [0, 1/\epsilon]^d$*. We will first observe that these two regions have large overlap:*

Lemma 1.4.3 *Let* $O^t = \{c^{<t} + [0, 1/\epsilon]^d\} \cap \{c^{<t+1} + [0, 1/\epsilon]^d\}$*. Then:*

$$\Pr[p^t \notin O^t] \leq C\epsilon \quad \Pr[\tilde{p}^t \notin O^t] \leq C\epsilon$$

Where $C = \max_{c \in \mathcal{C}} ||c||_1$

Proof 6 *We can calculate:*

$$\begin{aligned}
\Pr[p^t \notin O^t] &= \Pr[(c^{<t} + N^t) \notin \{c^{<t+1} + [0, 1/\epsilon]^d\}] \\
&= \Pr[N^t \notin \{c^t + [0, 1/\epsilon]^d\}] \\
&\leq \sum_{i=1}^{d} \Pr[N_i^t \notin [c_i^t, 1/\epsilon + c_i^t]] \\
&\leq \sum_{i=1}^{d} |c_i^t|\epsilon \\
&= ||c^t||_1 \epsilon \\
&\leq C\epsilon
\end{aligned}$$

The claim for $\tilde{p}^t$ follows identically.

The key observation is that because p^t and $\tilde{p}^t$ are uniformly distributed in their respective ranges, conditionally on lying in O^t, they are uniformly distributed within O^t. Since the corresponding actions a^t and $\tilde{a}^t$ taken by Be the Perturbed Leader and Follow the Perturbed Leader are deterministic post-processings of $\tilde{p}^t$ and p^t respectively, this means that the expected loss of a^t and $\tilde{a}^t$ are identical conditional on $p^t \in O^t$ and $\tilde{p}^t \in O^t$.

Lemma 1.4.4 *At each round t:*

$$\mathbb{E}[\langle a^t, c^t \rangle] \leq \mathbb{E}[\langle \tilde{a}^t, c^t] + 2CR\epsilon$$

Where $C = \max_{c \in \mathcal{C}} ||c||_1$ and $R = \max_{a \in \mathcal{A}, c \in \mathcal{C}} |\langle a, c \rangle|$

Proof 7 *Fix any round t. We can calculate:*

$$\begin{aligned}
\mathbb{E}[\langle a^t, c^t \rangle] &= \Pr[p^t \in O^t]\, \mathbb{E}[\langle a^t, c^t \rangle | p^t \in O^t] + \Pr[p^t \notin O^t]\, \mathbb{E}[\langle a^t, c^t \rangle | p^t \notin O^t] \\
&\leq \Pr[p^t \in O^t]\, \mathbb{E}[\langle a^t, c^t \rangle | p^t \in O^t] + C\epsilon \max_{a \in \mathcal{A}, c \in \mathcal{C}} \langle a, c \rangle \\
&= \Pr[p^t \in O^t]\, \mathbb{E}[\langle \tilde{a}^t, c^t \rangle | \tilde{p}^t \in O^t] + CR\epsilon \\
&\leq \mathbb{E}[\langle \tilde{a}^t, c^t \rangle | \tilde{p}^t \in O^t] + CR\epsilon \\
&\leq \mathbb{E}[\langle \tilde{a}^t, c^t \rangle] + 2CR\epsilon
\end{aligned}$$

Here we have used Lemma 1.4.3 twice.

We are now ready to complete the proof by chaining together our Lemmas. We have that the regret of Follow the Perturbed Leader to any action a is bounded by:

$$\begin{aligned}
\sum_{t=1}^{T} \left(\mathbb{E}[\langle a^t, c^t \rangle] - \langle a, c^t \rangle \right) &\leq \sum_{t=1}^{T} \left(\mathbb{E}[\langle \tilde{a}^t, c^t \rangle] - \langle a, c^t \rangle \right) + 2CRT\epsilon \\
&\leq \frac{A}{\epsilon} + 2CRT\epsilon
\end{aligned}$$

Here the first inequality follows from Lemma 1.4.4 and the second inequality follows from Lemma 1.4.2.

1.5 Follow the Regularized Leader and Online Gradient Descent

In this section we take another perspective on "Follow the Leader" style algorithms, and derive another online linear optimization algorithm that will sometimes be useful. We'll start by directly proving a regret bound for "Follow the Leader" (Algorithm 3). Recall that we have already seen a sequence on which Follow the Leader accumulates linear regret, so the bound that we prove cannot possibly guarantee sublinear regret for all sequences — but can give us insight into *what* kinds of sequences cause Follow the Leader to have large regret, and how to avoid it.

Theorem 4 *On any sequence of costs, the regret of Follow the Leader to any fixed action $a \in \mathcal{A}$ can be bounded as:*

$$\sum_{t=1}^{T} \left(\langle a^t, c^t \rangle - \langle a, c^t \rangle \right) \leq \sum_{t=1}^{T} \left(\langle a^t, c^t \rangle - \langle a^{t+1}, c^t \rangle \right)$$

Remark 1.5.1 *A couple of things about Theorem 4 are worth noting. First, the right hand side can be bounded as a function of the number of rounds the leader* changes *— i.e. the number of rounds t such that $a^t \neq a^{t+1}$. So if the "leader" is relatively stable, then Follow the Leader in fact already will have low regret. This is consistent with the bad example we say for Follow the Leader in Section 1.4 — in that example, the leader changed at every round, which can make the above bound grow linearly with T.*

Observe also that a^{t+1}, the action that Follow the Leader plays at round $t+1$, is also the action that the hypothetical algorithm "Be the Leader" (which has one round-lookahead) considered in Section 1.4 plays at round t.

Proof 8 (Proof of Theorem 4) *We've actually already proven this theorem! By subtracting off $\sum_{t=1}^{T} \langle a^t, c^t \rangle$ from both sides, we see that the statement we want to prove is that:*

$$\sum_{t=1}^{T} \langle a^{t+1}, c^t \rangle \leq \sum_{t=1}^{T} \langle a, c^t \rangle$$

But this is what we proved in Lemma 1.4.1 — that the one-lookahead-algorithm "Be the Leader" has non-positive regret.

Theorem 4 gives us an idea of how to modify Follow the Leader to give it a worst-case regret bound: "regularize" it so that it can't change actions too dramatically. This is morally quite similar to our solution in Follow the Perturbed Leader —- the added perturbations in FTPL make sure that a^t and a^{t+1} are *distributed* similarly — here we will instead make sure that

they are close in distance. Follow the Regularized Leader will be defined by a *regularization* function $\Psi : \mathcal{A} \to \mathbb{R}$.

Algorithm 5 Follow the Regularized Leader

for $t = 1$ to T **do**

Select the action:

$$a^t = \arg\min_{a \in \mathcal{A}} \left(\sum_{s=1}^{t-1} \langle a, c^s \rangle + \Psi(a) \right)$$

Theorem 5 *For any sequence of costs and any non-negative regularizer Ψ, the regret of Follow the Regularized Leader to any fixed action $a \in \mathcal{A}$ can be bounded as:*

$$\sum_{t=1}^{T} \left(\langle a^t, c^t \rangle - \langle a, c^t \rangle \right) \leq \sum_{t=1}^{T} \left(\langle a^t, c^t \rangle - \langle a^{t+1}, c^t \rangle \right) + (\Psi(a) - \Psi(a^1))$$

Proof 9 *We can view Follow the Regularized Leader as Follow the Leader in which there is a round 0 in which the cost for playing each action a is $\Psi(a)$. We can therefore apply Theorem 4 (whose proof did not require that the losses be linear) including round 0, which gives:*

$$(\Psi(a^0) - \Psi(a)) + \sum_{t=1}^{T} \left(\langle a^t, c^t \rangle - \langle a, c^t \rangle \right) \leq \sum_{t=1}^{T} \left(\langle a^t, c^t \rangle - \langle a^{t+1}, c^t \rangle \right) + (\Psi(a^0) - \Psi(a^1))$$

which gives the theorem after rearranging.

Remark 1.5.2 *Noting that $a^1 \in \arg\min_a \Psi(a)$, this implies a bound of:*

$$\sum_{t=1}^{T} \left(\langle a^t, c^t \rangle - \langle a, c^t \rangle \right) \leq \sum_{t=1}^{T} \left(\langle a^t, c^t \rangle - \langle a^{t+1}, c^t \rangle \right) + (\max_{a'} \Psi(a') - \min_{a'} \Psi(a'))$$

Therefore the goal is to find a regularizer Φ that doesn't take values that are too large, but also constrains a^t to be close to a^{t+1}. There are different interesting regularizers you could pick, but we'll investigate what happens when $\Phi(a) = \frac{1}{\eta}||a||_2^2$, the (scaled) squared Euclidean norm of a. In this case, when $\mathcal{A}$ is unconstrained, we can compute a closed form expression for a^t:

Lemma 1.5.1 *When $\mathcal{A} = \mathbb{R}^d$ and $\Phi(a) = \frac{1}{\eta}||a||_2^2$, then:*

$$a^t = -\frac{\eta}{2} \sum_{s=1}^{t-1} c^s$$

Proof 10 *By definition:*

$$a^t = \arg\min_a \left(\sum_{s=1}^{t-1} \langle a, c^s \rangle + \frac{1}{\eta} ||a||_2^2 \right)$$

a^t is minimizing a strictly convex function, and so to find the unique minimizer of this function, we can take the gradient and set it to zero. Doing so we find that a^t must solve:

$$\frac{2}{\eta} a^t = -\sum_{s=1}^{t-1} c^s$$

or in other words:

$$a^t = -\frac{\eta}{2} \sum_{s=1}^{t-1} c^s$$

Remark 1.5.3 *Observe that the form of the action a^t derived in Lemma 1.5.1 has a simple update rule: $a^{t+1} = a^t - \frac{\eta}{2} c^t$. In other words, the algorithm simply takes a step in the direction away from the gradient of the most recent loss function. As a result, we can call this algorithm "online gradient descent".*

Algorithm 6 Online Gradient Descent

Let $a^1 = 0$
for $t = 1$ to T **do**
  Select the action a^t.
  Observe c^t and let:

$$a^{t+1} = a^t - \frac{\eta}{2} c^t$$

Theorem 6 *For any sequence of costs $c^1, \ldots, c^T$ with $||c^t||_2 \leq C$ for all t, and for any $a \in \mathbb{R}^d$ with $||a||_2 \leq A$, Online Gradient Descent obtains regret:*

$$\sum_{t=1}^{T} \langle a^t, c^t \rangle \leq \sum_{t=1}^{T} \langle a, c^t \rangle + T \frac{C^2 \eta}{2} + \frac{A^2}{\eta}$$

Remark 1.5.4 *Setting $\eta = \frac{A}{C} \sqrt{\frac{2}{T}}$ we get a regret bound of:*

$$\sum_{t=1}^{T} \langle a^t, c^t \rangle \leq \sum_{t=1}^{T} \langle a, c^t \rangle + AC\sqrt{2T}$$

Observe that this bound is independent of the dimension of the actions and costs d, and depends only on their norm.

Proof 11 (Proof of Theorem 6) *We know from Lemma 1.5.1 that Online Gradient Descent is an instantiation of Follow the Regularized Leader with* $\Psi(a) = \frac{1}{\eta}||a||_2^2$*. Thus we can apply Theorem 5 to conclude that:*

$$\sum_{t=1}^{T} \left(\langle a^t, c^t\rangle - \langle a, c^t\rangle\right) \leq \sum_{t=1}^{T} \left(\langle a^t, c^t\rangle - \langle a^{t+1}, c^t\rangle\right) + \frac{1}{\eta}||a||_2^2$$

We have that $a^t - a^{t+1} = \frac{\eta}{2}c^t$*, and so*

$$||a^t - a^{t+1}||_2 \leq \frac{\eta}{2}||c^t||_2 \leq \frac{C\eta}{2}$$

Therefore we have that:

$$\langle a^t, c^t\rangle - \langle a^{t+1}, c^t\rangle = \langle (a^t - a^{t+1}), c^t\rangle \leq ||a^t - a^{t+1}|| \cdot ||c^t|| \leq \frac{C^2\eta}{2}$$

Thus we have the final regret bound:

$$\sum_{t=1}^{T} \left(\langle a^t, c^t\rangle - \langle a, c^t\rangle\right) \leq T\frac{C^2\eta}{2} + \frac{A^2}{\eta}$$

In general, Follow the Regularized Leader is a flexible design template: As an exercise, you can derive a Multiplicative-Weights like algorithm and bound by using the negative entropy regularizer $\Psi(a) = -\frac{1}{\eta}\sum_{i=1}^{d} a_i \ln(1/a_i)$.

1.6 Online Convex Optimization

We've now seen three algorithms for solving the online linear optimization problem (we were explicit about this for the Follow the Perturbed Leader and Online Gradient Descent algorithms, but its not hard to see that the Multiplicative Weights algorithm solves the online linear optimization problem where the Learner's action space is the Simplex). In fact, algorithms that solve the online linear optimization problem can also be used to solve the more general online convex optimization problem. First we will recall some basic definitions that we will need.

Definition 5 *A function* $f : \mathbb{R}^d \to \mathbb{R}$ *is* convex *if for all* $x_1, x_2 \in \mathbb{R}^d$*, and for all* $0 \leq \alpha \leq 1$*:*

$$f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha)f(x_2)$$

Linear functions are a special case of convex functions in which the inequality always holds with equality.

Definition 6 *A function $f : \mathbb{R}^d \to \mathbb{R}$ is* L-Lipschitz *in the L_1 norm if for all $x_1, x_2 \in \mathbb{R}^d$:*

$$|f(x_1) - f(x_2)| \leq L||x_1 - x_2||_1$$

If f is L-Lipschitz for some L we simply say that f is Lipschitz-continuous.

Definition 7 *Fix a convex function $f : \mathbb{R}^d \to \mathbb{R}$. A vector $c \in \mathbb{R}$ is a subgradient of f at $x \in \mathbb{R}$ if for all $x' \in \mathbb{R}$:*

$$f(x') - f(x) \geq \langle c, x' - x \rangle$$

If f is differentiable, then the gradient $\nabla f(x) = c$ is always a subgradient of f at x.

The following fact (which is not hard to prove, but we will take as given so as not to be led too far astray) will be useful in conceptualizing the reduction that follows in this section. It states that convex Lipschitz functions have bounded gradients, and vice versa.

Lemma 1.6.1 *A convex function f is L-Lipschitz (in the L_1 norm) if and only if for every x in its domain, and for every subgradient c of f at x, $||c||_\infty = \max_{i \in [d]} |c_i| \leq L$.*

Finally, let us define the online convex optimization problem.

Definition 8 (Online Convex Optimization) *In the L-Lipschitz online convex optimization problem:*

1. *The Learner has a convex action space $\mathcal{A} \subseteq \mathbb{R}^d$, and*
2. *The Adversary has an action space $\mathcal{C}$ consisting of L-lipschitz convex functions $\ell : \mathcal{A} \to \mathbb{R}^d$,*
3. *At each round t, the learner chooses an action $a^t \in \mathcal{A}$ and the adversary chooses a loss function $\ell^t \in \mathcal{C}$. The learner experiences cost $c_L^t = \ell^t(a^t)$.*

After realizing a transcript π^T, the regret that the learner experiences to action $a \in \mathcal{A}$ is:

$$Reg(\pi^T, a) = \sum_{t=1}^{T} \left(\ell^t(a^t) - \ell^t(a) \right)$$

Remark 1.6.1 *Here we have assumed that the Learner's action space $\mathcal{A}$ is convex. If it is not, as in previous sections, we can take the Learner's action space to be $\Delta\mathcal{A}$, the convex hull of $\mathcal{A}$, which is realized through randomization. Thus the online convex optimization setting generalizes the settings we have considered so far.*

We will observe a generic reduction that converts an arbitrary algorithm for online linear optimization (like multiplicative weights, online gradient descent,

or follow the perturbed leader) into an algorithm for online (Lipschitz) convex optimization with similar regret bounds.

Algorithm 7 A Reduction from Online Convex Optimization to Online Linear Optimization

Given: An algorithm LinearLearn for d-dimensional online linear optimization.
for $t = 1$ to T **do**
From LinearLearn, obtain action a^t.
From the adversary, obtain L-Lipschitz loss function ℓ^t.
Let $c^t \in [0, L]^d$ be a subgradient of ℓ^t at a^t (if ℓ^t is differentiable, $c^t = \nabla \ell^t(a^t)$.
Feed c^t to LinearLearn as a cost vector.

Theorem 7 *Suppose LinearLearn is an online linear optimization algorithm that obtains regret to each action $a \in \mathcal{A}$ bounded by $R(T)$ after T rounds, for all sequences of cost vectors $c^t \in [0, L]^d$. The the reduction in Algorithm 7 obtains regret to all fixed actions bounded as:*

$$Reg(\pi^T, a) = \sum_{t=1}^{T} \left(\ell^t(a^t) - \ell^t(a)\right) \leq R(T)$$

for all sequences of L-Lipschitz convex functions.

Proof 12 *Fix a comparison action $a \in \mathcal{A}$. We know from the guarantees of the online linear optimization algorithm that:*

$$\begin{aligned} R(T) &\geq \sum_{t=1}^{T} \left(\langle a^t, c^t\rangle - \langle a, c^t\rangle\right) \\ &= \sum_{t=1}^{T} \langle a^t - a, c^t\rangle \\ &\geq \sum_{t=1}^{T} \ell^t(a^t) - \ell^t(a) \\ &= Reg(\pi^T, a) \end{aligned}$$

Here the last inequality follows from the fact that c^t is a subgradient of ℓ^t at a^t and Definition 7.

Thus, for any fixed action space $\mathcal{A}$ and Lipschitz parameter L, we can use (e.g.) Multiplicative Weights, Online Gradient Descent, or Follow the Perturbed Leader to solve the online convex optimization problem with regret bounds scaling as $O(\sqrt{T})$.

Remark 1.6.2 *Finally, we note that although we have been discussing online convex* minimization*, all of these algorithms can be used for online concave* maximization*, simply by flipping the sign of the loss functions ℓ. If ℓ is convex, then $-\ell$ is concave, and a minimizer of ℓ within $\mathcal{A}$ is a maximizer of $-\ell$ within $\mathcal{A}$.*

Bibliographic Notes and Further Reading

Multiplicative weights is a classic algorithm with a long history [Littlestone, 1988, Littlestone and Warmuth, 1994, Freund and Schapire, 1999] — for a through introduction to several variants of the algorithm and its many applications see Arora et al. [2012]. A follow the perturbed leader like algorithm was developed by Hannan [1957]. Follow the Perturbed Leader, as developed in this chapter (applied to online linear optimization) is due to Kalai and Vempala [2005]. The online convex optimization framework and the first analysis of online gradient descent is due to Zinkevich [2003]. The follow the regularized leader framework was introduced in Abernethy et al. [2008]. Our treatment of follow the regularized leader draws from Orabona's notes here.

2

Zero Sum Games and the Minimax Theorem

CONTENTS

In this chapter we turn from sequential decision making to a seemingly unrelated topic: optimal play in zero-sum games. But as we will see, these two topics are very tightly linked, and in a strong sense, the kinds of algorithms we developed in Chapter 1 for online linear and convex optimization are constructive versions of the "minimax theorem", the fundamental property of zero sum games. We will see a two-way connection: we will prove the minimax theorem using the existence of online learning algorithms, and similarly will show how we can derive online learning algorithms from first principles starting from the minimax theorem.

2.1 Zero Sum Games

A zero-sum game is a strictly competitive game played between two players. We model this by assigning both players *action sets*, and defining a utility function that one player wants to maximize, and the other player wants to minimize. We will call these the *maximization* player ("Max") and the *minimization* player ("Min") respectively.

Definition 9 *A zero sum game is defined by:*

1. A Maximization *player endowed with a closed bounded action set* $\mathcal{A}_{max} \subset \mathbb{R}^m$*,*

2. A Minimization *player endowed with a closed bounded action set* $\mathcal{A}_{min} \subset \mathbb{R}^n$*, and*

3. A bounded utility function $u : \mathcal{A}_{max} \times \mathcal{A}_{min} \to [0, B]$.

How should Max and Min play a zero sum game? Lets start with an easy scenario: Suppose Max already knows what action $a_2 \in \mathcal{A}_{\min}$ that Min is going to play. In this case, he should play the action a_1 that maximizes the utility given Min's action:

$$a_1 \in \arg\max_{a \in \mathcal{A}_{\max}} u(a, a_2)$$

Similarly, if Min already knows what action $a_1 \in \mathcal{A}_{\max}$ that Max is going to play, she should play the action a_2 that *minimizes* the utility given Max's actions:

$$a_2 \in \arg\min_{a \in \mathcal{A}_{\min}} u(a_1, a)$$

We call these *best responses* for Max and Min respectively:

Definition 10 *The set of best responses for Max given an action* $a_2 \in \mathcal{A}_{min}$ *for Min is:*

$$Br_{max}(a_2) = \arg\max_{a \in \mathcal{A}_{max}} u(a, a_2)$$

The set of best responses for Min given an action $a_1 \in \mathcal{A}_{max}$ *for Max is:*

$$Br_{min}(a_1) = \arg\min_{a \in \mathcal{A}_{min}} u(a_1, a)$$

Remark 2.1.1 *The fact that we have assumed that the action spaces* $\mathcal{A}_{max}$ *and* $\mathcal{A}_{min}$ *are closed and bounded implies that the best response sets are well defined.*

If Max and Min are playing a pair of actions (a_1, a_2), and either player is not playing a best response to their opponent's action, then they will wish to change their action to a best response. If neither player wishes to change their action, we call this pair of actions an *equilibrium.*

Definition 11 *A pair of actions* $(a_1, a_2) \in \mathcal{A}_{max} \times \mathcal{A}_{min}$ *are a* Nash Equilibrium *if both:*

$$a_1 \in Br_{max}(a_2) \qquad a_2 \in Br_{min}(a_1)$$

2.2 From Sequential Decision Making to The Minimax Theorem

An equilibrium is a fixed point of simultanious play, and so in general its not clear how to go about computing one. Lets start with what seems like an

easier problem: How should Max play if he must first commit to his action a_1 and announce it to Min, who will then get an opportunity to best respond? Knowing that Min will play a best response, he should anticipate this, and play so as to *maximize* his utility after Min chooses her action to *minimize it*. That is, he should play a "maximin" strategy. Similarly, if Min must go first and commit to her action and let Max best respond, she should play a "minimax" strategy:

Definition 12 *a_1 is a* maximin *strategy for Max if:*

$$a_1 \in \arg\max_{a \in \mathcal{A}_{max}} \min_{a_2 \in \mathcal{A}_{min}} u(a, a_2)$$

Similarly, a_2 is a minimax *strategy for Min if:*

$$a_2 \in \arg\min_{a \in \mathcal{A}_{min}} \max_{a_1 \in \mathcal{A}_{max}} u(a_1, a)$$

We can similarly think about he minimax and maximin *values* of the game: what utility Max and Min respectively can guarantee if they must commit to their strategies up front and announce them:

Definition 13 *The minimax value of a game $v_{minimax}$ is:*

$$v_{minimax} = \min_{a_2 \in \mathcal{A}_{min}} \max_{a_1 \in \mathcal{A}_{max}} u(a_1, a_2)$$

The maximin value of a game $v_{maximin}$ is:

$$v_{maximin} = \max_{a \in \mathcal{A}_{max}} \min_{a_2 \in \mathcal{A}_{min}} u(a, a_2)$$

In a zero sum game, it can only be a disadvantage to go first, intuitively because you are revealing information to your opponent, without restricting her action space. This means that the minimax value of the game can only be larger than the maximin value:

Lemma 2.2.1 *In any zero sum game:*

$$v_{minimax} \geq v_{maximin}$$

Proof 13 *Let a_1 and a_2 be maximin and minimax strategies for Max and Min respectively. Then:*

$$v_{minimax} = \max_{a \in \mathcal{A}_{max}} u(a, a_2) \qquad v_{maximin} = \min_{a \in \mathcal{A}_{min}} u(a_1, a)$$

So we have:

$$\begin{aligned} v_{minimax} &= \max_{a \in \mathcal{A}_{max}} u(a, a_2) \\ &\geq u(a_1, a_2) \\ &\geq \min_{a \in \mathcal{A}_{min}} u(a_1, a) \\ &= v_{maximin} \end{aligned}$$

The fundamental fact about zero sum games is that (subject to some regularity conditions on the game), going first is not in fact a disadvantage: in particular, $v_{\text{maximin}} = v_{\text{minimax}}$. The regularity conditions that are needed are exactly those that allow players to use online convex optimization algorithms to obtain diminishing regret to the best action in their action space.

Theorem 8 *Fix a zero-sum game such that:*

1. The strategy sets $\mathcal{A}_{min} \subseteq \mathbb{R}^n$ and $\mathcal{A}_{max} \subset \mathbb{R}^m$ are closed, bounded, and convex,

2. The utility function $u : \mathcal{A}_{max} \times \mathcal{A}_{min} \to [0, B]$ satisfies:

(a) For all $a_2 \in \mathcal{A}_{min}$, $u(\cdot, a_2)$ is concave and Lipschitz continuous in its first argument, and

(b) For all $a_1 \in \mathcal{A}_{max}$, $u(a_1, \cdot)$ is convex and Lipschitz continuous in its second argument.

Then:

$$\max_{a_1 \in \mathcal{A}_{max}} \min_{a_2 \in \mathcal{A}_{min}} u(a_1, a_2) = \min_{a_2 \in \mathcal{A}_{min}} \max_{a_1 \in \mathcal{A}_{max}} u(a_1, a_2)$$

Remark 2.2.1 *The simplest kind of zero sum games involve finite action spaces A_1, A_2. Finite action spaces are not convex, and so to apply the minimax theorem, it is necessary to convexity them by letting players use* probability distributions *over their actions: $\mathcal{A}_{max} = \Delta A_1$, $\mathcal{A}_{min} = \Delta A_2$. When we do this, we extend the utility function u from the domain $A_1 \times A_2$ to the domain $\Delta A_1 \times \Delta A_2$ by defining for any pair $(p_1, p_2) \in \Delta A_1 \times \Delta A_2$, $u(p_1, p_2) = \mathbb{E}_{a_1 \sim p_1, a_2 \sim p_2}[u(a_1, a_2)]$. This function is* linear *in p_1 and p_2 (by linearity of expectation), and so convex and concave in each argument.*

Proof 14 (Proof of Theorem 8) *We know from Lemma 2.2.1 that $v_{minimax} \geq v_{maximin}$. Suppose for point of contradiction that the inequality is strict, and let $\epsilon = v_{minimax} - v_{maximin} > 0$.*

We now imagine repeated play of the game between Min and Max for T rounds, which generates a sequence of action pairs $\{(a^t, b^t)\}_{t=1}^T$. Since for every $a^t \in \mathcal{A}_{max}$, $u(a^t, \cdot)$ is Lipschitz and convex in its second argument, we will let Min choose her action every day using an online convex optimization algorithm (like Multiplicative Weights or Follow the Perturbed Leader, using the reduction from Theorem 7). We will let Max best respond to Min's action each day: $a^t \in Br_{max}(b^t)$; we will then feed Min's online convex optimization algorithm the loss function $\ell^t(b^t) = u(a^t, b^t)$, which provides the feedback she needs to select b^{t+1}. Let us now analyze the cumulative utility of the game. We know two things: First, because Min's action space is convex and bounded, and the loss functions $\ell^t(b^t) = u(a^t, b^t)$ we feed to Min's learning algorithm are convex and Lipschitz, she has an $O(\sqrt{T})$ regret guarantee to the best fixed action in hindsight (the constants in the bound depend on things like the Lipschitz constant L and the diameter of the action space, so we elide them with

Big-O notation):

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T} u(a^t, b^t) &\leq \min_{b \in \mathcal{A}_{min}} \frac{1}{T}\sum_{t=1}^{T} u(a^t, b) + O\left(\frac{1}{\sqrt{T}}\right) \\ &\leq \min_{b \in \mathcal{A}_{min}} u\left(\frac{1}{T}\sum_{t=1}^{T} a^t, b\right) + O\left(\frac{1}{\sqrt{T}}\right) \\ &\leq v_{maximin} + O\left(\frac{1}{\sqrt{T}}\right)\end{aligned}$$

Here, in the second inequality we use the fact that for every $b \in \mathcal{A}_{min}$, $u(\cdot, b)$ is concave in its first argument, and apply Jensen's inequality. We also use the fact that $\mathcal{A}_{max}$ is a convex set, which implies that $\left(\frac{1}{T}\sum_{t=1}^{T} a^t\right) \in \mathcal{A}_{max}$.

On the other hand, since for every t, $a^t \in Br_{max}(b^t)$, we also have that:

$$u(a^t, b^t) \geq v_{minimax}$$

Combining these two bounds gives:

$$v_{minimax} \leq \frac{1}{T}\sum_{t=1}^{T} u(a^t, b^t) \leq v_{maximin} + O\left(\frac{1}{\sqrt{T}}\right)$$

Thus by taking T to be sufficiently large we can obtain:

$$v_{minimax} < v_{maximin} + \epsilon$$

which contradicts our initial assumption. Thus we must have that $v_{minimax} = v_{maximin}$, proving the theorem.

The Minimax theorem (Theorem 8) allows us to speak of the *value* v of a zero sum game — we don't have to specify whether we mean the minimax or maximin value, as they are the same.

Definition 14 *In any zero-sum game for which the conditions of the minimax Theorem hold, we define the* value *of the game as the unique value $v \in \mathbb{R}$ such that:*

$$v_{minimax} = v_{maximin} = v$$

A simple consequence of the minimax theorem is that Nash equilibria in zero sum games are pairs of minimax/maximin strategies.

Lemma 2.2.2 *Fix a zero sum game in which the Minimax theorem holds. A pair of strategies $(a_1, a_2) \in \mathcal{A}_{max} \times \mathcal{A}_{min}$ is a Nash equilibrium if and only if a_1 is a maximin strategy and a_2 is a minimax strategy.*

Proof 15 *Let v be the value of the zero sum game (whose existence is guaranteed by the minimax theorem). First suppose that a_1 and a_2 are maximin and minimax strategies respectively. We must have that $u(a_1, a_2) \geq v$. and $u(a_1, a_2) \leq v$, and so $u(a_1, a_2) = v$. Thus we have that $a_1 \in Br_{max}(a_2)$ (since a_2 is a minimax strategy), and $a_2 \in Br_{min}(a_1)$ (since a_1 is a maximin strategy), implying that (a_1, a_2) are a Nash equilibrium.*

Next, suppose that (a_1, a_2) are a Nash equilibrium. Since $a_1 \in Brmax(a_2)$, and since there is a maximin strategy in $\mathcal{A}_{max}$, we must have that $u(a_1, a_2) \geq v$. Similarly, since $a_2 \in Br_{min}(a_1)$, we must have that $u(a_1, a_2) \leq v$ and so $u(a_1, a_2) = v$. Since a_2 is a best response to a_1 and vice versa, we have that a_1 and a_2 are maximin and minimax strategies respectively.

2.3 Computing Minimax Strategies

So far we've used the existence of online convex optimization algorithms to prove the minimax theorem, but sometimes it will be useful for us to actually be able to compute approximate minimax (and maximin) equilibria in particular zero sum games. We can use online convex optimization algorithms to do this as well. There are a few variants, each of which is sometimes useful. First we define an approximate minimax and maximin equilibrium:

Definition 15 *Fix a zero sum game $(\mathcal{A}_{max}, \mathcal{A}_{min}, u)$. An ϵ-approximate maximin strategy is an action $a \in \mathcal{A}_{max}$ such that for all $b \in \mathcal{A}_{min}$:*

$$u(a, b) \geq \min_{b' \in \mathcal{A}_{min}} u(a, b') - \epsilon$$

Similarly, an ϵ-approximate minimax strategy is an action $b \in \mathcal{A}_{min}$ such that for all $a \in \mathcal{A}_{max}$:

$$u(a, b) \leq \max_{a' \in \mathcal{A}_{max}} u(a', b) + \epsilon$$

Our algorithms will be of two types, both of which simulate repeated play of the zero sum game over some number of rounds T. Either we will play two no-regret/online convex optimization algorithms against one another, or we will have one player play using an online convex optimization algorithm, and the other player "best respond" at each round. In fact, we don't necessarily need the "best response" player to play a best response — its enough if they play a strategy that achieves the value of the game at each round (i.e. the do not need to be able to exploit their opponent if their opponent is playing badly).

Definition 16 *Fix a zero sum game satisfying the conditions of the minimax theorem, and let v be the value of the game. A* value oracle *for Max is a*

mapping $Val : \mathcal{A}_{min} \to \mathcal{A}_{max}$ *such that for all* $b \in \mathcal{A}_{min}$, $Val(b) = a$ *such that:*

$$u(a, b) \geq v$$

Similarly a value oracle for Min is a mapping $Val : \mathcal{A}_{max} \to \mathcal{A}_{min}$ *such that for all* $a \in \mathcal{A}_{max}$, $Val(a) = b$ *such that:*

$$u(a, b) \leq v$$

Remark 2.3.1 *Note that computing an actual best response —* $Val(b) = Br_{max}(b)$ *for Max, and* $Val(a) = Br_{min}(a)$ *for Min gives a value oracle, but it might sometimes be easier to implement a value oracle than to compute the best response in a game.*

Algorithm 8 Computing a Minimax Equilibrium: Value Oracle vs. No Regret

Given: A zero sum game $(\mathcal{A}_{\max}, \mathcal{A}_{\min}, u)$ satisfying the conditions of the minimax theorem, a Value oracle $Val : \mathcal{A}_{\min} \to \mathcal{A}_{\max}$ for Max, an online convex optimization algorithm OnlineConvex operating over action space $\mathcal{A}_{\min}$ and loss space $\{\ell = u(a, \cdot)\}_{a \in \mathcal{A}_{\max}}$ that promises regret $R(T)$ to every action $b \in \mathcal{A}_{\min}$ after T rounds, and an approximation parameter ϵ.
Let T be such that $R(T)/T \leq \epsilon$
for $t = 1$ to T **do**
  Get action b^t from OnlineConvex,
  Let $a^t = Val(b^t)$
  Feed loss $\ell^t = u(a^t, \cdot)$ to OnlineConvex.
Let $\bar{a} = \frac{1}{T}\sum_{t=1}^{T} a^t$
Return $\bar{a}$

Theorem 9 *The action* $\bar{a}$ *output by Algorithm 8 is an* ϵ*-approximate Maximin strategy.*

Proof 16 *By the regret bound of OnlineConvex we know for every* $b \in \mathcal{A}_{min}$*:*

$$\begin{aligned} \epsilon &\geq \frac{1}{T}\sum_{t=1}^{T}\left(u(a^t, b^t) - u(a^t, b)\right) \\ &\geq \frac{1}{T}\left(\sum_{t=1}^{T} u(a^t, b^t)\right) - u(\bar{a}, b) \end{aligned}$$

where the last inequality follows from the fact that $u(\cdot, b^*)$ *is convex in its first argument and Jensen's inequality.*

Rearranging we have that for all $b \in \mathcal{A}_{min}$*:*

$$\begin{aligned} u(\bar{a}, b) &\geq \frac{1}{T}\left(\sum_{t=1}^{T} u(a^t, b^t)\right) - \epsilon \\ &\geq v - \epsilon \end{aligned}$$

where the last inequality follows from the fact that a^t is selected by a value oracle $Val(b^t)$, and so for all t, $u(a^t, b^t) \geq v$ Thus $\bar{a}$ is an ϵ-approximate Maximin strategy.

Remark 2.3.2 *Observe that we didn't really need an* exact *value oracle. If the guarantee was that for $a^t = Val(b^t)$, $u(a^t, b^t) \geq v - \epsilon$, then we would have found a 2ϵ-approximate Maximin strategy, so an approximate Value oracle is enough.*

Note also that the situation is symmetric: If we want to compute an ϵ-approximate Minimax strategy for Min, we can reverse the role of the players and have Max play a no regret algorithm and have Min play according to an (approximate) value oracle.

There is another general way to compute Minimax and Maximin strategies in a zero-sum game: have *both* players play the game for T rounds so that they have at most ϵ average regret. They could do this by both playing according to online convex optimization algorithms, or we could play an online convex optimization algorithm against a best response (not just a value) oracle.

Theorem 10 *Fix a zero sum game $(\mathcal{A}_{max}, \mathcal{A}_{min}, u)$ satisfying the conditions of the minimax theorem, and let $\{(a^t, b^t)\}_{t=1}^T$ be a sequence of action pairs that mutually have ϵ average regret with respect to each other — i.e. such that:*

$$\frac{1}{T} \min_{b \in \mathcal{A}_{min}} \sum_{t=1}^{T} u(a^t, b) + \epsilon \geq \frac{1}{T} \sum_{t=1}^{T} u(a^t, b^t) \geq \frac{1}{T} \max_{a \in \mathcal{A}_{max}} \sum_{t=1}^{T} u(a, b^t) - \epsilon$$

Let

$$\bar{a} = \frac{1}{T} \sum_{t=1}^{T} a^t \qquad \bar{b} = \frac{1}{T} \sum_{t=1}^{T} b^t$$

Then $\bar{a}$ is a 2ϵ-approximate maximin strategy and $\bar{b}$ is a 2ϵ-approximate minimax strategy.

Proof 17 *We prove the claim about $\bar{a}$: The claim about $\bar{b}$ follows exactly symmetrically.*

From the right hand side of the no regret guarantee together with the convexity of u in its second argument (and Jensen's inequality), we have that:

$$\begin{aligned} \frac{1}{T} \sum_{t=1}^{T} u(a^t, b^t) &\geq \max_{a \in \mathcal{A}_{max}} \frac{1}{T} \sum_{t=1}^{T} u(a, b^t) - \epsilon \\ &\geq \max_{a \in \mathcal{A}_{max}} u(a, \bar{b}) - \epsilon \\ &\geq \min_{b \in \mathcal{A}_{min}} \max_{a \in \mathcal{A}_{max}} u(a, b) - \epsilon \\ &= v - \epsilon \end{aligned}$$

From the left hand side of the no-regret guarantee together with the concavity of u in its first argument (and Jensen's inequality) we have that for every $b \in \mathcal{A}_{min}$:

$$\begin{aligned} u(\bar{a}, b) &\geq \frac{1}{T}\sum_{t=1}^{T} u(a^t, b) \\ &\geq \frac{1}{T}\sum_{t=1}^{T} u(a^t, b^t) - \epsilon \\ &\geq v - 2\epsilon \end{aligned}$$

Which establishes that $\bar{a}$ is a 2ϵ-approximate maximin strategy.

2.4 From the Minimax Theorem to Sequential Decision Making

We used the existence of online convex optimization algorithms with regret guarantees to prove the minimax theorem for zero sum games. But historically, the minimax theorem came first. Suppose we knew the minimax theorem: could we use it to derive the existence of online convex optimization algorithms with regret guarantees? The answer is *yes*, and so in a strong sense, these kinds of "no regret learning" algorithms should be viewed as equivalent, constructive versions of the minimax theorem.

Recall from Section 1.6 that to derive online convex optimization algorithms it suffices to derive online *linear* optimization algorithms, and that the simplest kind of online linear optimization algorithm is an algorithm (like multiplicative weights) that selects amongst k actions at each round — this is just online linear optimization in which the learner's action space is the k-dimensional probability simplex. So that's what we will do for simplicity — we'll see how to derive a multiplicative weights like algorithm and regret bound in this way.

First we recall the setting: there are k actions, and cost vectors $c^t \in [0,1]^k$. At every round, the learner chooses some (distribution on) action(s) $p^t \in \Delta\mathcal{A}$, after which an adversary chooses a cost vector c^t. The cost for the algorithm at round t if they play an action i is $c_L^t = c_i^t$ — the expected cost for the algorithm at round t is $\mathbb{E}_{i\sim p^t}[c_L^t] = \langle p^t, c^t \rangle$. After T rounds, recall that the accumulated regret to action i is:

$$\text{Reg}(\pi^T, i) = C_L^T - C_i^T = \sum_{t=1}^{T}(c_L^t - c_i^t)$$

And the overall regret is $\mathrm{Reg}(\pi^T) = \max_{i\in[k]} \mathrm{Reg}(\pi^T, i)$. Our high level strategy will be to invoke the minimax theorem to find a strategy p^t that the learner can play at each round t to minimize the increase in her overall regret —- but the max term that shows up in the overall regret of the learner makes this increase a complicated, non convex-concave function of p^t and c^t, which prevents a direct invocation. Instead we need a *surrogate* function that has better analytic properties, but can be used to upper-bound overall regret. A natural choice is the softmax function, which smoothly approximates the max function using exponentials.

Definition 17 *At round T, define the softmax surrogate with parameter η to be:*

$$L(\pi^T) = \sum_{i=1}^{k} \exp\left(\eta Reg(\pi^T, i)\right)$$

We can use the softmax surrogate to upper bound the overall regret of the algorithm as follows:

Lemma 2.4.1 *For all T:*

$$Reg(\pi^T) \leq \frac{1}{\eta} \ln(L(\pi^T))$$

Proof 18

$$\begin{aligned}
\eta Reg(\pi^T) &= \max_{i\in[k]} \eta Reg(\pi^T, i) \\
&= \ln\left(\exp\left(\max_{i\in[k]} \eta Reg(\pi^T, i)\right)\right) \\
&= \ln\left(\max_{i\in[k]} \exp\left(\eta Reg(\pi^T, i)\right)\right) \\
&\leq \ln\left(\sum_{i=1}^{k} \exp\left(\eta Reg(\pi^T, i)\right)\right) \\
&= \ln(L(\pi^T))
\end{aligned}$$

Dividing by η gives the result.

Remark 2.4.1 *Observe that the soft-max upper bound on regret is reasonably tight — it cannot over-estimate the regret by more than an additive term of $\frac{1}{\eta} \cdot \log k$. Because $L(\pi^T) \leq k \exp(\eta(Reg(\pi^T)))$, we have that $\frac{1}{\eta}\log(L(\pi^T)) \leq \frac{\log k}{\eta} + Reg(\pi^T)$*

Thus it will suffice to design an algorithm that can control the growth of $L(\pi^T)$.

Definition 18 *Fix a transcript π^{s-1}. Given an action $i \in [k]$ and a cost vector $c \in [0,1]^k$ let $\tilde{\pi}^s = \pi^{s-1} \circ (i,c)$, the continuation of the transcript that would result if the learner picked action i and the adversary picked cost vector c. Let:*

$$\Delta_2^{\pi^{s-1}}(c,i) = L(\tilde{\pi}^s) - L(\pi^{s-1})$$

be the change in the squared error surrogate that would result from the play (i,c) at round s.

Our first step is to analytically upper bound the increase in the softmax surrogate that results from playing action i against cost vector c at round s:

Lemma 2.4.2 *For any $\eta \leq 1$:*

$$\Delta^{\pi^{s-1}}(c,i) \leq \left(\sum_{j=1}^{k} \exp\left(\eta Reg(\pi^{s-1}, j)\right) \eta(c_i - c_j) \right) + \eta^2 L(\pi^{s-1})$$

Proof 19

$$\begin{aligned}
\Delta^{\pi^{s-1}}(c,i) &= L(\tilde{\pi}^s) - L(\pi^{s-1}) \\
&= \sum_{j=1}^{k} \exp\left(\eta Reg(\tilde{\pi}^s, j)\right) - \exp\left(\eta Reg(\pi^{s-1}, j)\right) \\
&= \sum_{j=1}^{k} \exp\left(\eta Reg(\pi^{s-1}, j) + \eta(c_i - c_j)\right) - \exp\left(\eta Reg(\pi^{s-1}, j)\right) \\
&= \sum_{j=1}^{k} \exp\left(\eta Reg(\pi^{s-1}, j)\right) \left(\exp\left(\eta(c_i - c_j)\right) - 1\right) \\
&\leq \sum_{j=1}^{k} \exp\left(\eta Reg(\pi^{s-1}, j)\right) \left(\eta(c_i - c_j) + (\eta(c_i - c_j))^2\right) \\
&\leq \sum_{j=1}^{k} \exp\left(\eta Reg(\pi^{s-1}, j)\right) \left(\eta(c_i - c_j) + \eta^2\right) \\
&= \left(\sum_{j=1}^{k} \exp\left(\eta Reg(\pi^{s-1}, j)\right) \eta(c_i - c_j) \right) + \eta^2 L(\pi^{s-1})
\end{aligned}$$

Where the second to last inequality follows from the fact that for any $x \leq 1$, $\exp(x) \leq 1 + x + x^2$ and the last inequality follows from the assumption that $c_i, c_j \in [0,1]$.

Observe that by construction, for a transcript π^T generated by a sequence of plays (p^t, c^t), the expected softmax surrogate regret is exactly:

$$\mathbb{E}[L(\pi^T)] = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{i \sim p^t} [\Delta^{\pi^{t-1}}(c^t, i)]$$

Thus, to guarantee that the expected regret of the Learner is small, our goal will be to find distributions p^t at each round t that guarantee that $\mathbb{E}_{i\sim p^t}[\Delta^{\pi^{t-1}}(c,i)]$ is small for all c (remembering that we don't know the relevant loss vector c^t at the time that we must pick p^t).

Towards this end, we define a zero-sum game between the learner and the adversary at each round t as follows. We will identify the Learner with the minimization player and the Adversary with the maximization player.

Definition 19 *The round-t softmax-surrogate game is*

1. The learner's action space is $\mathcal{A}_{min} = \Delta[k]$, which is a convex set.

2. The adversary's action space is $\mathcal{A}_{max} = [0,1]^k$, which is a convex set.

3. For each $c \in \mathcal{A}_{max}$ and $p \in \mathcal{A}_{min}$, the utility function is defined as:

$$u(c,p) = \mathop{\mathbb{E}}_{i\sim p}\left[\sum_{j=1}^{k} \exp\left(\eta Reg(\pi^{s-1},j)\right)\eta(c_i - c_j)\right]$$

This is a bounded utility function that is linear (and hence convex/concave) in both of its arguments (as at round t, $Reg(\pi^{t-1},j)$ is simply a fixed constant).

We observe that this game satisfies the conditions of the minimax Theorem 8.

Lemma 2.4.3 *The maximin value of the round t softmax-surrogate game is:*

$$\max_{c\in[0,1]^k}\min_{p\in\Delta[k]} u(c,p) \leq 0$$

Proof 20 *For any $c \in \mathcal{A}_{max}$, let $i^*(c) \in \arg\min_{i\in[k]} c_i$ be a coordinate of minimum cost. the Learner has a best response $p^*(c)$ corresponding to a distribution that places all of its weight on $i^*(c)$. We have:*

$$u(c,p^*(c)) = \sum_{j=1}^{k} \exp\left(\eta Reg(\pi^{s-1},j)\right)\eta(c_{i^*(c)} - c_j) \leq 0$$

Where the inequality follows because term by term, $\exp\left(\eta Reg(\pi^{s-1},j)\right) \geq 0$ because of the non-negativity of the exponential function, and $(c_{i^*(c)} - c_j) \leq 0$ because by definition of $i^*(c)$, $c_{i^*(c)} \leq c_j$ for all j.

Since the conditions of the minimax Theorem (Theorem 8) are satisfied by our round-t surrogate softmax game, we can swap the min and the max and conclude:

Lemma 2.4.4 *The minimax value of the round t softmax surrogate game is:*

$$\min_{p\in\Delta[k]}\max_{c\in[0,1]^k} u(c,p) \leq 0$$

In other words, at each round t, there exists a distribution over actions $p \in \Delta[k]$ such that for *all* cost vectors $c \in [0,1]^k$ that the adversary might choose, $u(c,p) \leq 0$.

Because we have defined the utility function in our round-t softmax-surrogate game to be a value that we can use to upper bound the per-round expected change in softmax-surrogate regret, we can immediately use this fact to derive a regret bound on Algorithm 9, which simply plays at every round a distribution $p^t \in \Delta[k]$ such that: $\max_{c \in [0,1]^k} u(c,p) \leq 0$. The existence of such a distribution is guaranteed by Lemma 2.4.4.

Algorithm 9 A Minimax Based Sequential Decision Making Algorithm

for $t = 1$ to T **do**
  Construct the round t softmax-surrogate game as a function of π^{t-1} with utility function u.
  Play $p^t \in \Delta[k]$ such that: $\max_{c \in [0,1]^k} u(c,p) \leq 0$.

Theorem 11 *Against any sequence of cost functions $c^1, \ldots, c^T \in [0,1]^k$, Algorithm 9 has expected regret bounded by:*

$$\mathop{\mathbb{E}}_{\pi^T}[Reg(\pi^T)] \leq \frac{\ln k}{\eta} + T\eta$$

Choosing $\eta = \sqrt{\frac{\ln k}{T}}$ gives:

$$\mathop{\mathbb{E}}_{\pi^T}[Reg(\pi^T)] \leq 2\sqrt{T \ln k}$$

Remark 2.4.2 *Note that this exactly matches the worst-case regret bound we proved for the Multiplicative Weights algorithm in Theorem 2!*

Proof 21 *We start by upper bounding the expected softmax surrogate regret. From the definition of $\Delta^{\pi^{s-1}}(c,i)$ and Lemma 2.4.2 we have that for all rounds s:*

$$\begin{aligned}
\mathop{\mathbb{E}}_{p^s}[L(\pi^s)|\pi^{s-1}] &= L(\pi^{s-1}) + \mathop{\mathbb{E}}_{i \sim p^s}[\Delta^{\pi^{s-1}}(c^s, i)] \\
&\leq (1+\eta^2)L(\pi^{s-1}) + \mathop{\mathbb{E}}_{i \sim p^s}\left[\left(\sum_{j=1}^{k} \exp\left(\eta Reg(\pi^{s-1}, j)\right)\eta(c_i - c_j)\right)\right] \\
&= (1+\eta^2)L(\pi^{s-1}) + u(c^s, p^s) \\
&\leq (1+\eta^2)L(\pi^{s-1})
\end{aligned}$$

Where the last inequality follows from the fact (justified by Lemma 2.4.4) that p^s satisfies $u(c^s, p^s) \leq 0$ for all values of c^s.

Observing that $L(\pi^0) = k$ and applying the above bound inductively, we find that:

$$\mathop{\mathbb{E}}_{\pi^T}[L(\pi^T)] \leq k(1+\eta^2)^T \leq k\exp(T\eta^2)$$

We can now apply Lemma 2.4.1 to bound the regret by the softmax surrogate regret.

$$\begin{aligned}
\mathop{\mathbb{E}}_{\pi^T}[Reg(\pi^T)] &\leq \mathop{\mathbb{E}}_{\pi^T}\left[\frac{1}{\eta}\ln(L(\pi^T))\right] \\
&\leq \frac{1}{\eta}\ln\left(\mathop{\mathbb{E}}_{\pi^T}\left[L(\pi^T)\right]\right) \\
&\leq \frac{1}{\eta}\left(\ln k + T\eta^2\right) \\
&= \frac{\ln k}{\eta} + T\eta
\end{aligned}$$

We proved the existence of the strategy used at each round of 9 non-constructively using the minimax theorem. How would we actually implement it? We require finding a minimax strategy for the softmax surrogate game at each round to find a p^t such that $\max_{c\in[0,1]^k} u(c,p) \leq 0$. In general we could do this using linear programming, or the no-regret dynamics approaches to computing minimax optimal strategies that we studied in Section 2.3. But in this case, it turns out that there is a simple closed form expression for p^t!

Lemma 2.4.5 *Let $p^t \in \Delta[k]$ be such that:*

$$p_i^t = \frac{1}{\Phi^t}\exp(\eta Reg(\pi^{t-1}, i)) \qquad \Phi^t = \sum_i \exp(\eta Reg(\pi^{t-1}, i))$$

Then p^t satisfies the requirements of Algorithm 9 — for all $c \in [0,1]^k$:

$$u(c, p^t) = \mathop{\mathbb{E}}_{i\sim p^t}\left[\left(\sum_{j=1}^k \exp\left(\eta Reg(\pi^{t-1}, j)\right)\eta(c_i - c_j)\right)\right] \leq 0$$

Proof 22 *We can directly compute:*

$$\begin{aligned}
u(c, p^t) &= \mathop{\mathbb{E}}_{i\sim p^t}\left[\left(\sum_{j=1}^k \exp\left(\eta Reg(\pi^{t-1}, j)\right)\eta(c_i - c_j)\right)\right] \\
&= \sum_{i=1}^k p_i^t \sum_{j=1}^k \exp\left(\eta Reg(\pi^{t-1}, j)\right)\eta(c_i - c_j) \\
&= \frac{1}{\Phi^t}\sum_{i=1}^k\sum_{j=1}^k \exp(\eta Reg(\pi^{t-1}, i))\exp\left(\eta Reg(\pi^{t-1}, j)\right)\eta(c_i - c_j) \\
&= \frac{\eta}{\Phi^t}\sum_{i=1}^k\sum_{j=1}^k \exp(\eta Reg(\pi^{t-1}, i))\exp\left(\eta Reg(\pi^{t-1}, j)\right)c_i - \\
&\quad \frac{\eta}{\Phi^t}\sum_{i=1}^k\sum_{j=1}^k \exp(\eta Reg(\pi^{t-1}, i))\exp\left(\eta Reg(\pi^{t-1}, j)\right)c_j \\
&= 0
\end{aligned}$$

Thus we have derived a concrete, easy to implement algorithm, that is a slight variant of multiplicative weights —this variant is sometimes called "Exponential Weights" .

Algorithm 10 The Exponential Weights Algorithm

For each action $i \in \mathcal{A}$, set $w_i^1 = 1$. Let $W^1 = \sum_{i \in \mathcal{A}} w_i^1$.
for $t = 1$ to T **do**
Play the distribution p^t defined as:

$$p_i^t = \frac{w_i^t}{W^t}$$

Observe costs c^t and update weights such that for each $i \in \mathcal{A}$:

$$w_i^{t+1} = w_i^t \exp(-\eta c_i^t) \qquad W^{t+1} = \sum_{i \in \mathcal{A}} w_i^{t+1}$$

Theorem 12 *The Exponential Weights Algorithm (Algorithm 10) implements Algorithm 9, and so satisfies the regret bound proven in Theorem 11:*

$$\mathop{\mathbb{E}}_{\pi^T}[Reg(\pi^T)] \leq \frac{\ln k}{\eta} + T\eta$$

Choosing $\eta = \sqrt{\frac{\ln k}{T}}$ *gives:*

$$\mathop{\mathbb{E}}_{\pi^T}[Reg(\pi^T)] \leq 2\sqrt{T \ln k}$$

Proof 23 *From the update rule, we can compute that at round* t*, Algorithm 10 plays a distribution* p_i^t *defined as*

$$p_i^t = \frac{1}{W^t} \exp\left(-\eta \sum_{s=1}^{t-1} c_i^s\right) \qquad W^t = \sum_{i=1}^{k} \exp\left(-\eta \sum_{s=1}^{t-1} c_i^s\right)$$

Our goal is to show that this distribution is identical to the distribution that we proved in Lemma 2.4.5 implements Algorithm 9. Recall that we wrote that distribution as:

$$q_i^t = \frac{1}{\Phi^t} \exp(\eta Reg(\pi^{t-1}, i)) \qquad \Phi^t = \sum_i \exp(\eta Reg(\pi^{t-1}, i))$$

Consider any coordinate i *of the distribution defined in Lemma 2.4.5. we*

can calculate:

$$\begin{aligned} q_i^t &= \frac{\exp(\eta Reg(\pi^{t-1}, i))}{\sum_j \exp(\eta Reg(\pi^{t-1}, j))} \\ &= \frac{\exp\left(\eta \sum_{s=1}^{t-1}(c_L^s - c_i^s)\right)}{\sum_j \exp\left(\eta \sum_{s=1}^{t-1}(c_L^s - c_j^s)\right)} \\ &= \frac{\exp\left(\eta \sum_{s=1}^{t-1} c_L^s\right) \cdot \exp\left(-\eta \sum_{s=1}^{t-1} c_i^s\right)}{\exp\left(\eta \sum_{s=1}^{t-1} c_L^s\right) \sum_j \exp\left(-\eta \sum_{s=1}^{t-1} c_j^s\right)} \\ &= \frac{\exp\left(-\eta \sum_{s=1}^{t-1} c_i^s\right)}{\sum_j \exp\left(-\eta \sum_{s=1}^{t-1} c_j^s\right)} \\ &= p_i^t \end{aligned}$$

Thus $p^t = q^t$, and so the result follows from Lemma 2.4.5 and Theorem 11.

Bibliographic Notes and Further Reading

The proof of the minimax theorem using no-regret learning algorithms is originally due to Freund and Schapire [1996], who also show how to compute minimax equilibria using no regret learning algorithms. The variant in which a no-regret learner is used to play against a "value oracle" is from Haghtalab et al. [2023a]. The derivation of exponential weights using the minimax theorem follows Lee et al. [2022].

3

Multi-Objective Sequential Learning

CONTENTS

So far we've viewed sequential learning as having a single goal: obtaining a diminishing regret guarantee, always as computed over the entire sequence. Similarly we have studied zero sum games in which there is a single, one dimensional objective function that one player wants to maximize and the other wants to minimize. In this case there was a tight connection between learning algorithms with regret guarantees and equilibrium play in games.

In this chapter, things will get more complicated: in a sequential learning setting, there might be more than one objective that we simultaneously want to control. We'll show how to use the tools we have developed for online convex optimization to solve this problem. We'll give a number of applications of this technique; but as a motivating example we'll think about computing approximate *correlated* equilibria, a solution concept in *general* sum games, in which there might be many players, each of which have different utility functions.

3.1 Motivating Example: Convergence to Correlated Equilibria

Here we will study general games, which remove some of the restrictions of zero sum games. In particular, there may now be many players in the game, and the players can have arbitrary (not necessarily strictly opposing) cost functions.

Definition 20 *A (general sum) game is defined by:*

1. *A finite set of n players $\{1, \ldots, n\}$*
2. *For each player, i, a closed, bounded action set $\mathcal{A}_i \subseteq \mathbb{R}^m$,*
3. *For each player i, a bounded cost function $c_i : \mathcal{A}_1 \times \ldots \times \mathcal{A}_n \to [0, B]$*

Remark 3.1.1 *For a game to have nice properties (the existence of various kinds of equilibria) we will generally also want each player's action sets $\mathcal{A}_i$ to be convex. As usual, if $\mathcal{A}_i$ is not already convex, we can convexify it by taking its convex hull, which corresponds to letting players choose probability distributions over their actions. These are often referred to as "mixed strategies" in game theory.*

General sum games do not in general enjoy the nice structure of the minimax theorem that we proved for Zero Sum games in Chapter 2. We can still define Nash equilibria — stable game states such that no player can decrease their cost by unilaterally changing their action. However, although Nash equilibria are guaranteed to exist when certain mild technical conditions are met, they are not in general easy to learn.

Definition 21 *A set of actions $a = (a_1, \ldots, a_n) \in \mathcal{A}_1 \times \ldots \times \mathcal{A}_n$ form a Nash equilibrium if for every player i:*

$$c_i(a) \leq \min_{a_i' \in \mathcal{A}_i} c_i(a_i', a_{-i})$$

Here (a_i', a_{-i}) denotes the vector of actions a in which the value of the i'th coordinate a_i has been replaced with a_i'.

We will state but not prove the existence of Nash equilibrium:

Theorem 13 *Fix a game in which the action sets $\mathcal{A}_i$ are all convex and closed, and the cost functions $c_i(a_i, a_{-i})$ are continuous in all of their arguments and convex in a_i. Then a Nash equilibrium exists.*

We will focus instead on a more computationally tractable solution concept, called a correlated equilibrium. A correlated equilibrium shares the same defining philosophy as a Nash equilibrium — defining a "stable state" such that no player can unilaterally deviate in a way that will improve their cost. The key distinction is that a correlated equilibrium allows players to randomize their actions using correlated randomness, and to consider deviations that are a function of their own portion of that randomness. Traditionally, one imagines that a correlated equilibrium is implemented using a "correlating device" like a traffic light, which supplies signals or suggested actions to each player. The players observe their own signals, and know how their signals are correlated with the signals shown to others, but do not directly observe other's signals.

Definition 22 *Let $\mathcal{P} \in \Delta(\mathcal{A}_1 \times \ldots \times \mathcal{A}_n)$ be a distribution over actions. $\mathcal{P}$ is an ϵ-approximate correlated equilibrium of a game if for all players i and all deviation functions $\phi_i : \mathcal{A}_i \to \mathcal{A}_i$:*

$$\mathop{\mathbb{E}}_{a\sim\mathcal{P}}[c_i(a)] \leq \mathop{\mathbb{E}}_{a\sim\mathcal{P}}[c_i(\phi_i(a_i), a_{-i})] + \epsilon$$

Note that in a correlated equilibrium, players need not consider only deviations to single fixed actions a'_i, but may consider more complex deviations $\phi(a_i)$ that are *functions* of their suggested part of the correlated play a_i, deviating to different actions depending on what they are suggested to play. Implicit in this is a notion of timing in the game: players *first* see their signal/suggested action, and *then* are free to decide what action to play. They may use information that they learned from their signal to decide on what action to play. This is closely related to a more demanding kind of "regret" in sequential play than we have seen before:

Definition 23 *Fix an action space for a learner $\mathcal{A}$ and a class Φ of "action modification rules" $\phi : \mathcal{A} \to \mathcal{A}$. Given a sequence of action distributions and loss functions $\pi^T = \{(p^1, \ell^1), \ldots, (p^T, \ell^T)\}$, the learner's regret to a strategy modification rule $\phi \in \Phi$ is:*

$$Reg(\pi^T, \phi) = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{a^t\sim p^t} \left[\left(\ell^t(a^t) - \ell^t(\phi(a^t))\right)\right]$$

We say that the learner has Φ-regret bounded by α if:

$$\max_{\phi\in\Phi} Reg(\pi^T, \phi) \leq \alpha$$

If Φ is the set of all *action modification rules $\phi : \mathcal{A} \to \mathcal{A}$ we refer to Φ regret as* swap *regret.*

Remark 3.1.2 *If $\Phi = \{\phi_a\}_{a\in\mathcal{A}}$ is the set of all* constant *action modification rules defined such that for all $a' \in \mathcal{A}$, $\phi_a(a') = a$, then Φ-regret is referred to as* external *regret and corresponds to the kind of regret we have encountered previously: regret to the best* fixed *action in hindsight.*

The connection between swap regret and correlated equilibrium is straightforward. First, we define swap regret in the context of a general sum game, in which at every round, each of the n players plays i an action a_i^t. The loss with respect to which regret is defined is simply the cost function for that player: $\ell_i^t = c_i(a_i^t, a_{-i}^t)$.

Definition 24 *Fix a general sum game and a sequence of mixed strategy profiles $P^1, \ldots, P^T$ where each P^t is a vector of action distributions $P^t \in \Delta\mathcal{A}_1 \times \ldots \times \Delta\mathcal{A}_n$. This sequence has swap regret α if for every player i:*

$$\max_{\phi:\mathcal{A}_i \to \mathcal{A}_i} \mathop{\mathbb{E}}_{a^t \sim P^t} \left[\sum_{t=1}^{T} \left(c_i(a_i^t, a_{-i}^t) - c_i(\phi(a_i^t), a_{-i}^t) \right) \right] \leq \alpha$$

Lemma 3.1.1 *Fix a general sum game and a sequence of mixed strategy profiles $P^1 \ldots, P^T$ that has swap regret α. Let $\mathcal{P}$ be the distribution that first samples P^t uniformly from the set of mixed strategy profiles $\{P^1, \ldots, P^T\}$ and then samples action profile $a^t \sim P^t$. Then $\mathcal{P}$ is an ϵ-approximate correlated equilibrium for $\epsilon = \frac{\alpha}{T}$.*

Proof 24 *This follows from the definitions. For all players i and action modification rules $\phi : \mathcal{A}_i \to \mathcal{A}_i$:*

$$\begin{aligned}
\mathop{\mathbb{E}}_{a \sim \mathcal{P}}[c_i(a)] &= \frac{1}{T} \sum_{t=1}^{T} \mathop{\mathbb{E}}_{a^t \sim P^t}[c_i(a^t)] \\
&\leq \frac{1}{T} \left(\sum_{t=1}^{T} \mathop{\mathbb{E}}_{a^t \sim P^t}[c_i(\phi(a_i^t), a_{-i}^t)] + \alpha \right) \\
&= \mathop{\mathbb{E}}_{a \sim \mathcal{P}}[c_i(\phi(a_i), a_{-i})] + \frac{\alpha}{T}
\end{aligned}$$

Thus, if we can derive learning algorithms that guarantee this stronger form of swap regret at rates that grow sublinearly with T, we will have an algorithm (and a learning dynamic) that converges to approximate correlated equilibrium. In the next section we'll develop a general framework that will allow us to derive such an algorithm as a special case (among many other applications).

3.2 A General Framework for Multiobjective Sequential Learning

In this section we derive a general framework for optimizing multiple objectives at once in a sequential learning setting. This will have many applications; one category of these applications is deriving algorithms that have stronger kinds

of regret guarantees than we have seen thus far, including the swap regret guarantees that we saw imply convergence to the set of correlated equilibria.

We consider a sequential decision making setting in which the learner has d distinct cost functions, each of which they would like to keep as small as possible over a sequential interaction. We define a very general setting, in which the action space for the learner and the adversary, as well as the nature of each of the loss functions can change at each round. We will not always need this generality, but when we do it will come in handy. In the general setting, the adversary gets to choose the action spaces at every round, as well as the "loss increment functions" for each of the d loss objectives. These can differ at each round. The learner then selects an action, the adversary responds, and loss acrues to each loss objective according to the loss increment function for that objective.

Definition 25 (The Multiobjective Optimization Game) *The d-objective optimization game proceeds in rounds $t = 1, \ldots, T$. At each round t:*

1. The adversary selects the environment for round t which comprises:

(a) Closed convex, finite dimensional action spaces $\mathcal{A}^t, \mathcal{B}^t \subset \mathbb{R}^m$ for the learner and adversary respectively, and

(b) A continuous loss increment function ℓ_i^t for each of the d losses $i \in [d]$. Each $\ell_i^t : \mathcal{A}^t \times \mathcal{B}^t \to [-1, 1]$ is convex in its first argument and concave in its second argument.

2. The Learner selects an action $a^t \in \mathcal{A}^t$ and reveals it to the adversary.

3. The Adversary selects an action $b^t \in \mathcal{B}^t$.

4. The Learner suffers and observes loss $\ell_i^t(a^t, b^t)$, accruing to each of the loss objectives $i \in [d]$.

At the end of T rounds, the cumulative loss that the learner suffers for each objective $i \in [d]$ is: $L_i^T = \sum_{t=1}^T \ell_i^t(a^t, b^t)$.

The goal of the learner in interacting within the multiobjective optimization game is to upper bound the maximum accumulated loss across all objectives: i.e. she wants to minimize $\max_i L_i^T$. What is a reasonable benchmark? Here we introduce the "Adversary Moves First"value for the environment at round t:

Definition 26 *Fix the environment at round t, defined by $(\mathcal{A}^t, \mathcal{B}^t, \{\ell_i^t\}_{i=1}^d)$. The adversary moves first value at round t is:*

$$v_A^t = \max_{b \in \mathcal{B}^t} \min_{a \in \mathcal{A}^t} \max_{i \in [d]} \ell_i^t(a, b)$$

Informally, the adversary moves first value is the smallest upper bound on the d loss increments at round t that the learner could guarantee, if she had the advantage of first observing the adversary's chosen action b^t before deciding on her own best response a^t. Note that the order of play here is reversed compared to how it actually proceeds in the multiobjective optimization game. In the actual interaction, the learner moves first and then best responds. It is tempting to appeal to the minimax theorem here to assert that the learner can do just as well in this play order, but this turns out not to be the case. Although the loss increments in each coordinate $\ell_i^t(a,b)$ are indeed convex/concave, the maximum over coordinates $\max_{i\in[d]} \ell_i^t(a,b)$ does not preserve concavity, and hence the conditions of the minimax theorem are not satisfied. Indeed, in this setting, the minimax theorem simply doesn't hold:

Example 1 *Suppose the action spaces for both the learner and the adversary are the d-dimensional simplex: $\mathcal{A} = \mathcal{B} = \Delta[d]$. Let the loss increment in coordinate i be $\ell_i(a,b) = (b_i - a_i)$, the difference between the weight that the adversary and the learner place on coordinate i. If the adversary moves first and plays b, the learner can best respond and play $a = b$, guaranteeing that the loss increment in every coordinate is equal to 0: thus the adversary moves first value $v_A = 0$ for this environment.*

On the other hand, suppose the learner moves first: for every vector $a \in \mathcal{A}$ that she might choose, there is a coordinate i such that $a_i \leq \frac{1}{d}$. The adversary can best respond by playing a vector b that places all of its weight on this coordinate i: $b_i = 1$. Hence $\ell_i(a,b) \geq 1 - \frac{1}{d}$. Thus there is a large gap between the adversary moves first value for this environment and the "learner moves first" value — the minimax theorem does not hold.

The above example shows that trying to obtain maximum loss equal to the adversary-moves-first value of the game in a 1-round interaction is impossible. Nevertheless, we will be able to *approach* the average of the adversary-moves-first values of the environments over a larger sequence of T interactions.

Definition 27 (Adversary Moves First Regret) *Fix a transcript of interaction in the multiobjective optimization game $\pi^T = \{(\mathcal{A}^t, \mathcal{B}^t, \{\ell_i^t\}_{i=1}^d), a^t, b^t\}_{t=1}^T$. The adversary moves first regret of this transcript is:*

$$Reg_{AMF}(\pi^T) = \max_{i\in[d]} L_i^T - \sum_{t=1}^T v_A^t = \max_{i\in[d]} \left(\sum_{t=1}^T \left(\ell_i^t(a^t, b^t) - v_A^t \right) \right)$$

Our goal will be to design algorithms that guarantee that the Adversary Moves First regret grows sublinearly with T. In most of our applications, we will define the loss increments so that the adversary moves first value of each environment is 0: $v_A^t = 0$ for all t. In this case, the adversary moves first regret is simply the maximum accumulated loss in any coordinate: $\max_i L_i^T$.

We give the algorithm in Algorithm 11, which is a reduction from the problem of guaranteeing AMF regret bounded by $R(T)$ to the problem (that

we have already solved) of selecting a distribution over d actions at every round to obtain cost that is as large as the cumulative cost of the highest cost action in hindsight, up to a regret bound of $R(T)$. (i.e. the online linear optimization problem over the probability simplex $\Delta[d]$). We've already seen how to solve this problem using several algorithms (e.g. multiplicative weights, online gradient descent, exponential weights, etc.). Note that we have generally framed online linear optimization as the problem of *minimizing* cumulative cost, whereas here it is more convenient to use an online linear optimization algorithm that *maximizes* cumulative cost — but we can obtain an online linear maximization algorithm from an online linear minimization algorithm simply by negating the cost vectors.

The intuition for the reduction is simple. We run a sequential linear optimization algorithm over distributions on a set of d actions, with each action corresponding to one of the loss objectives in the Multiobjective Optimization Game. From its regret bound, the sequential linear optimization algorithm is guaranteed to experience cumulative cost that is nearly as large as that of the action with largest cumulative cost in hindsight — which corresponds to the cumulative loss of the maximum of the d loss objectives in the Multiobjective Optimization Game. This is exactly the quantity that we wish to control. Thus if we want to upper bound the cumulative loss of the highest loss objective in the Multiobjective Optimization Game, it suffices to upper bound the cumulative loss of the sequential linear optimization algorithm. Of course we don't know what this is going to be, because we don't know what action the adversary will choose at each round: but what we can try to do is upper bound the loss of the sequential linear optimization algorithm in the worst case over the adversary's action. That is exactly what the reduction does: it defines a zero sum game in which the objective is the loss that the sequential linear optimization algorithm will experience, and plays a minimax strategy in that game. Thus the cumulative loss of the online linear optimization algorithm, and hence the cumulative loss of the maximum coordinate of the multiobjective optimization game, is upper bounded by the sum of the values of the zero-sum games we have defined along the way, which turn out to be exactly the Adversary Moves First values v_A^t of each round of the multiobjective optimization game.

Algorithm 11 Reduction from AMF to Simple Regret

Given A sequential linear maximization algorithm Alg operating over action space $\Delta[d]$ and accepting cost vectors in $[-1,1]^d$.

for $t = 1$ to T **do**

Obtain distribution p^t from Alg.

Define a zero-sum game in which the minimization player's action are $\mathcal{A}^t$, the maximization players actions are $\mathcal{B}^t$, and the utility function is:

$$u^t(a,b) = \sum_{i=1}^{d} p_i^t \ell_i^t(a,b)$$

Compute a minimax equilibrium strategy of this game a^t for the minimization player and select action a^t

Observe the adversary's action b^t and report cost vector c^t to Alg, defined such that in each coordinate $i \in [d]$:

$$c_i^t = \ell_i^t(a^t, b^t)$$

The result is that if we instantiate Algorithm 11 with an online linear optimization algorithm that has regret bound $R(T)$, then we will obtain an AMF regret bound of $R(T)$ as well in the multi-objective optimization problem!

Theorem 14 *Suppose Algorithm 11 is instantiated with a sequential linear optimization algorithm operating over $\Delta[d]$ that has the guarantee that for any sequence of losses of length T bounded in [-1,1], it generates a transcript π^T with regret at most $R(T)$:*

$$\max_{i \in [d]} Reg(\pi^T, i) \leq R(T)$$

Then in any d-objective optimization game, after T rounds, Algorithm 11 obtains AMF regret at most $R(T)$:

$$\max_{i \in [d]} \left(\sum_{t=1}^{T} \left(\ell_i^t(a^t, b^t) - v_A^t \right) \right) \leq R(T)$$

Proof 25 *From the regret bound of the online linear maximization algorithm, we know that for all $j \in [d]$:*

$$\begin{aligned} \sum_{t=1}^{T} \left(c_j^t - \langle p^t, c^t \rangle \right) &= \sum_{t=1}^{T} \left(\ell_j^t(a^t, b^t) - \sum_{i=1}^{d} p_i^t \ell_i^t(a^t, b^t) \right) \\ &\leq R(T) \end{aligned}$$

Or, rearranging:

$$\max_{j \in [d]} L_j^T = \max_{j \in [d]} \sum_{t=1}^{T} \ell_j^t(a^t, b^t) \leq \sum_{t=1}^{T} \sum_{i=1}^{d} p_i^t \ell_i^t(a^t, b^t) + R(T)$$

Algorithm 11 selects a^t at every round so that:

$$a^t \in \arg\min_{a\in\mathcal{A}^t}\max_{b\in\mathcal{B}^t}\sum_{i=1}^{d} p_i^t \ell_i^t(a,b)$$

Because for each coordinate i, we know that $\ell_i^t(a,b)$ is convex in a and concave in b, and linear combinations of convex/concave functions are convex/concave, the utility function $u^t(a,b) = \sum_{i=1}^{d} p_i^t \ell_i^t(a,b)$ satisfies the conditions of the minimax theorem (Theorem 8), and so we know that:

$$\begin{aligned}
\min_{a\in\mathcal{A}^t}\max_{b\in\mathcal{B}^t}\sum_{i=1}^{d} p_i^t \ell_i^t(a,b) &= \max_{b\in\mathcal{B}^t}\min_{a\in\mathcal{A}^t}\sum_{i=1}^{d} p_i^t \ell_i^t(a,b) \\
&\leq \max_{b\in\mathcal{B}^t}\min_{a\in\mathcal{A}^t}\max_{i\in[d]} \ell_i^t(a,b) \\
&= v_A^t
\end{aligned}$$

Hence we know that at every round t,

$$\sum_{i=1}^{d} p_i^t \ell_i^t(a^t,b^t) \leq v_A^t$$

Finally, this lets us conclude that for every loss coordinate j:

$$\begin{aligned}
L_j^T &= \max_{j\in[d]}\sum_{t=1}^{T} \ell_j^t(a^t,b^t) \\
&\leq \sum_{t=1}^{T}\sum_{i=1}^{d} p_i^t \ell_i^t(a^t,b^t) + R(T) \\
&\leq \sum_{t=1}^{T} v_A^t + R(T)
\end{aligned}$$

Or in other words, the AMF regret is bounded by $R(T)$.

Finally, we can instantiate Algorithm 11 with a particular online linear maximization algorithm over the simplex $\Delta[d]$ — the exponential weights algorithm we derived in Algorithm 10. The exponential weights algorithm has a particularly simple form for the weights p_i^t, and a good concrete regret bound. We give this instantiation below in Algorithm 12.

Algorithm 12 Multiobjective Optimization with Exponential Weights

for $t = 1$ to T **do**

Define the distribution $p^t \in \Delta[d]$ as:

$$p_i^t = \frac{\exp\left(\frac{\eta}{2}\sum_{t'=1}^{t-1} \ell_i^{t'}(a^{t'}, b^{t'})\right)}{\sum_{j=1}^{d} \exp\left(\frac{\eta}{2}\sum_{t'=1}^{t-1} \ell_j^{t'}(a^{t'}, b^{t'})\right)}$$

Define a zero-sum game in which the minimization player's action are $\mathcal{A}^t$, the maximization players actions are $\mathcal{B}^t$, and the utility function is:

$$u^t(a, b) = \sum_{i=1}^{d} p_i^t \ell_i^t(a, b)$$

Compute a minimax equilibrium strategy of this game a^t for the minimization player and select action a^t

Theorem 15 *In any d-objective optimization game, after T rounds, Algorithm 12 obtains AMF regret at most $4\sqrt{T \ln d}$:*

$$\max_{i \in [d]} \left(\sum_{t=1}^{T} \left(\ell_i^t(a^t, b^t) - v_A^t \right) \right) + 4\sqrt{T \ln d}$$

Proof 26 *We simply instantiate Theorem 14 with the regret bound proven for the exponential weights algorithm in Theorem 12. Note that there we stated the regret bound for Exponential Weights when the costs were scaled in* $[0, 1]$ *and the algorithm was* minimizing *cost. In our case, the costs are scaled in* $[-1, 1]$ *and we are* maximizing *cost. We simply apply the cost transformation described in Remark 1.3.2 and negate the cost vectors to obtain the given algorithm and regret bound.*

3.3 Controlling Regret on Multiple Subsequences

As our first application of online multiobjective optimization, we will design algorithms for choosing amongst k actions that have diminishing regret to the best action in hindsight — not just on average over the whole sequence, but simultaneously on many different subsequences, which may be defined by (among other things) the actions we choose to play. If we instantiate this for the k subsequences corresponding to the rounds in which we chose to play each of our k actions, this will correspond to a guarantee of no swap regret

as we defined it in Section 3.1, giving us learning algorithms that converge to correlated equilibrium when played against one another in general sum games.

First we introduce a general/abstract framework for subsequence regret. In our formulation, there will be d subsequences on which the learner will want to guarantee that their cost is comparable to the cost of the best action in hindsight. Whether or not each round t is included in a subsequence i will be determined by a subsequence selection function $E(t, a^t, x^t)$ which can depend on the round t, the action a^t chosen by the learner at that round, and x^t, which represents any additional context or outside information available to the learner before round t. This is expressive enough to define subsequences like "Rounds 1000 through 2000", "Rounds on which we play action 3", "Rounds on which the person with features x^t we are making a decision about is female", and combinations thereof. Although our language of "subsequence selection" is suggestive that each round will either be a part of a given subsequence or not (and most of our applications will be of this sort), we will define subsequence selection functions more generally so that they can take values in $[0, 1]$, which we can interpret as having the ability to fractionally select rounds to participate in each subsequence.

Definition 28 (Subsequence Selection and Regret) *Fix an action space $\mathcal{A} = [k]$ and a context space $\mathcal{X}$. Let $\mathcal{E}$ be a collection of m subsequence selection functions $E : [T] \times \mathcal{A} \times \mathcal{X} \to [0, 1]$. The interaction between the learner and the adversary proceeds in rounds $t = 1$ to T:*

1. *The learner observes a context $x^t \in \mathcal{X}$ (if any) that may be chosen by the adversary.*
2. *The learner chooses a distribution over actions $p^t \in \Delta\mathcal{A}$*
3. *The adversary chooses a cost vector $c^t \in [0, 1]^k$.*
4. *The learner samples experiences expected cost $\langle p^t, c^t \rangle$.*

Fix a transcript $\pi^T = \{(x^t, p^t, c^t)\}_{t=1}^T$. The expected regret to action $i \in [k]$ on subsequence $E \in \mathcal{E}$ is:

$$Reg(\pi^T, E, i) = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{j \sim p^t} \left[E(t, j, x^t) \left(c_j^t - c_i^t \right) \right]$$

The learner has expected $\mathcal{E}$-subsequence regret bounded by α if:

$$\max_{E \in \mathcal{E}, i \in [k]} Reg(\pi^T, E, i) \leq \alpha$$

Finally we introduce some terminology. If a collection of events $\mathcal{E}$ contains events E that are independent of the played action (i.e. such that for all $a, a' \in \mathcal{A}$, we always have $E(t, a, x) = E(t, a', x)$), then we say that $\mathcal{E}$ is an action independent *collection of events. When writing the subsequence selection functions E, we elide dependence on parameters that are not used: for example, if a subsequence depends only on the action chosen we will write $E(a)$ rather than $E(t, a, x)$, etc.*

Remark 3.3.1 *Observe that the "simple" regret we have mostly worked with thus far is just the special case of subsequence regret for the single subsequence that includes all rounds: $E(t) = 1$ for all t.*

We can cast the problem of obtaining no-subsequence regret over k actions and a collection of subsequences $\mathcal{E}$ of size $|\mathcal{E}| = m$ as a $d = (m \cdot k)$-objective optimization game.

Definition 29 (Subsequence Regret Multiobjective Optimization Game) *Fix a collection of m subsequence selection functions $\mathcal{E}$ and k actions $[k]$. We define a $d = m \cdot k$ multiobjective optimization game in which the environment at each round t is:*

> *1. The action space for the learner is $\mathcal{A}^t = \Delta[k]$ and the action space for the adversary is $\mathcal{B}^t = [0,1]^k$.*
>
> *2. The loss increment functions are defined as follows. For each subsequence indicator function $E \in \mathcal{E}$ and action $i \in [k]$, we define loss increment:*
>
> $$\ell^t_{E,i}(p^t, c^t) = \mathop{\mathbb{E}}_{j \sim p^t}\left[E(t, j, x^t)\left(c^t_j - c^t_i\right)\right]$$

We can immediately apply Theorem 15 to get a bound on any subsequence regret problem:

Theorem 16 *Fix a collection of m subsequence selection functions $\mathcal{E}$ and a set of k actions. If we run Algorithm 12 on the multiobjective optimization game defined in Definition 29, then against any sequence of costs, we generate a transcript π^T that has subsequence regret bounded by:*

$$\max_{E \in \mathcal{E}, i \in [k]} Reg(\pi^T, E, i) \leq 4\sqrt{T(\ln m + \ln k)}$$

Theorem 17 *We first verify that the construction in Definition 29 satisfies the conditions required of a multiobjective optimization game. Indeed, the action sets $\mathcal{A}^t, \mathcal{B}^t$ are closed and convex. The loss functions are linear in both $\mathcal{A}^t$ and $\mathcal{B}^t$ (and hence convex/concave), and bounded in $[-1,1]$ as required.*

Next, we verify that the adversary-moves-first value of the game at each round t is 0: $v^T_A = 0$. To see this, fix any action c for the adversary. Let $i^*(c) \in \arg\min_{i \in [k]} c_i$ be a minimum coordinate of the cost vector c. If we let $p(c)$ be the distribution that places all of its weight on action $i^*(c)$. We have that:

$$\begin{aligned}
v^t_A &= \max_{c \in \mathcal{B}^t} \min_{p \in \mathcal{A}^t} \max_{E \in \mathcal{E}, i \in [k]} \ell^t_{E,i}(p, c) \\
&\leq \max_{c \in \mathcal{B}^t, E \in \mathcal{E}, i \in [k]} \ell^t_{E,i}(p(c), c) \\
&= \max_{c \in \mathcal{B}^t, E \in \mathcal{E}, i \in [k]} E(t, i^*(c), x^t)(c_{i^*(c)} - c_i) \\
&\leq 0
\end{aligned}$$

where the last inequality follows from the fact that by definition of i^*, $c_{i^*(c)} \leq c_i$ *for all* i.

The bound then follows from Theorem 15. For every E, i*:*

$$\begin{aligned} Reg(\pi^T, E, i) &= \mathop{\mathbb{E}}_{j \sim p^t}\left[E(t,j,x^t)\left(c_j^t - c_i^t\right)\right] \\ &= \sum_{t=1}^{T} \ell_{E,i}^t(p^t, c^t) \\ &= \sum_{t=1}^{T} \left(\ell_{E,i}^t(p^t, c^t) - v_A^t\right) \\ &\leq 4\sqrt{T \ln(m \cdot k)} \end{aligned}$$

Remark 3.3.2 *Note that this is the same (!) bound we get for simple regret with Multiplicative Weights or Exponential Weights (Theorem 12), except the* $\ln k$ *term has been replaced with a* $\ln(mk)$ *term, where* m *is the number of subsequences we are interested in.*

So we have an extremely general technique for minimizing regret simultaneously across *any* collection of m subsequences, with regret bounds growing only logarithmically in m. But running Algorithm 12 generically requires computing a minimax equilibrium at each step to find the distribution p^t to play. In certain cases we can do better and get a closed form for p^t.

3.3.1 Action Independent Subsequences

Recall that a set of subsequence selection functions $\mathcal{E}$ is *action independent* if for every $E \in \mathcal{E}$, for every t, x, and for every $a, a' \in \mathcal{A}$: $E(t,a,x) = E(t,a',x)$. In other words, the subsequences do not depend on the actions chosen by the algorithm. We can write the subsequence selection functions as $E(t,x)$ in this case. Just as we did when deriving the exponential weights algorithm in Section 2.4, in this case we can derive a closed form for the minimax equilibrium that needs to be played at each round by Algorithm 12.

Algorithm 13 Getting Action Independent Subsequence Regret

Given A collection $\mathcal{E}$ of m action-independent subsequence selection functions.

for $t = 1$ to T **do**

Observe context x^t

Define the distribution $p^t \in \Delta[k]$ as:

$$p_i^t = \frac{\sum_{E\in\mathcal{E}} E(t,x^t)\cdot \exp\left(\frac{\eta}{2}\sum_{t'=1}^{t-1} E(t',x^{t'})\left(\langle p^{t'},c^{t'}\rangle - c_i^{t'}\right)\right)}{\sum_{j=1}^{k}\sum_{E\in\mathcal{E}} E(t,x^t)\cdot \exp\left(\frac{\eta}{2}\sum_{t'=1}^{t-1} E(t',x^{t'})\left(\langle p^{t'},c^{t'}\rangle - c_j^{t'}\right)\right)}$$

Play distribution p^t.

Remark 3.3.3 *The algorithm has a simple form: For each subsequence E, it computes a weight proportional to the exponential of the regret to action i on that subsequence, and sums up the weights, scaling each one by $E(t,x^t)$, the degree to which subsequence E is active at round t. Then it gives action i weight proportional to this sum.*

For binary *subsequences (in which each round t is either contained in the subsequence $E(t,x^t)=1$ or not $E(t,x^t)=0$, it simply zeros out the weight for each of the subsequences that are inactive at round t.*

Theorem 18 *For any collection $\mathcal{E}$ of m action independent subsequence selection functions, Algorithm 13 implements Algorithm 12 for the Subsequence Regret Multiobjective Optimization Game (Definition 29), and hence obtains the regret bound from Theorem 16:*

$$\max_{E\in\mathcal{E},i\in[k]} Reg(\pi^T,E,i) \le 4\sqrt{T(\ln m + \ln k)}$$

Proof 27 *Algorithm 12 needs to play a distribution p^t at each round that is a minimax equilibrium for the game with utility function defined as:*

$$\begin{aligned} u^t(p,c) &= \sum_{E\in\mathcal{E},i\in[k]} \frac{\exp\left(\frac{\eta}{2}\sum_{t'=1}^{t-1}\ell_{E,i}^{t'}(p^{t'},c^{t'})\right)}{\sum_{E',j}\exp\left(\frac{\eta}{2}\sum_{t'=1}^{t-1}\ell_{E',j}^{t'}(p^{t'},c^{t'})\right)}\ell_{E,i}^t(p,c) \\ &= \sum_{E\in\mathcal{E},i\in[k]} \frac{\exp\left(\frac{\eta}{2}\sum_{t'=1}^{t-1}E(t,x^{t'})\left(\langle p^{t'},c^{t'}\rangle - c_i^{t'}\right)\right)}{\sum_{E',j}\exp\left(\frac{\eta}{2}\sum_{t'=1}^{t-1}E'(t,x^t)\left(\langle p^{t'},c^{t'}\rangle - c_j^{t'}\right)\right)} E(t,x^t)\left(\langle p,c\rangle - c_i\right) \end{aligned}$$

This game has value 0*. Thus we need to show that for the distribution p^t defined in Algorithm 13, for all cost vectors c, $u^t(p^t,c)\le 0$.*

Observe from the way we have defined p^t, can write $u^t(p,c)$ as:

$$\begin{aligned} u^t(p,c) &= \sum_{i=1}^{k} \frac{1}{\Phi^t} p_i^t \left(\langle p, c\rangle - c_i\right) \\ &= \frac{1}{\Phi^t} \sum_{i=1}^{k} p_i^t \left(\sum_{j=1}^{k} p_j c_j - c_i \right) \\ &= \frac{1}{\Phi^t} \left(\sum_{i=1}^{k} \sum_{j=1}^{k} p_i^t p_j c_j - \sum_{i=1}^{k} \sum_{j=1}^{k} p_i^t c_i \right) \\ &= \frac{1}{\Phi^t} \sum_{i=1}^{k} \sum_{j=1}^{k} \left(c_j (p_i^t p_j - p_j^t) \right) \end{aligned}$$

where Φ^t is a normalization factor defined as

$$\Phi^t = \frac{\sum_{E',j} \exp\left(\frac{\eta}{2} \sum_{t'=1}^{t-1} E'(t, x^t) \left(\langle p^{t'}, c^{t'} \rangle - c_j^{t'} \right)\right)}{\sum_{E',j} E'(t, x^t) \exp\left(\frac{\eta}{2} \sum_{t'=1}^{t-1} E'(t, x^t) \left(\langle p^{t'}, c^{t'} \rangle - c_j^{t'} \right)\right)}$$

So, plugging in $p = p^t$ we have that for all c:

$$\begin{aligned} u^t(p^t, c) &= \frac{1}{\Phi^t} \sum_{i=1}^{k} \sum_{j=1}^{k} \left(c_j (p_i^t p_j^t - p_j^t) \right) \\ &\leq 0 \end{aligned}$$

Since for every i, j: $p_i^t p_j \leq p_j^t$.

So what are some interesting examples of action-independent subsequence regret? We briefly explore a few:

3.3.1.1 Adaptive Regret

The basic regret guarantees we have proven roughly speaking promise the following: On average, over the sequence of rounds $1, \ldots, T$, the cost of the algorithm is at most the cost of the best (lowest cost) fixed action in hindsight — up to a regret bound of $O(\sqrt{T})$. But the best fixed action in hindsight is the action that has the lowest cost after all T rounds, and if we start keeping track of regret at some round $t' > 0$ — or stop keeping track of regret at some round $t < T$, then we do not necessarily have the same guarantees. On the other hand, an *adaptive* regret guarantee asks that simultaneously for all $0 < t' < t \leq T$, our algorithm has low regret to the best fixed action as evaluated on the subsequence $[t', t]$.

Definition 30 (Adaptive Regret) *Fix an action space $\mathcal{A} = [k]$ and a transcript $\pi^T = \{p^s, c^s\}_{s=1}^T$. The expected regret to action $i \in [k]$ on the subsequence $[t', t]$ for $0 < t' < t \leq T$ is:*

$$Reg(\pi^T, [t', t], i) = \sum_{s=t'}^{t} (\langle p^s, c^s \rangle - c_i^s)$$

The learner has expected adaptive regret bounded by α if:

$$\max_{i \in [k], 0 < t' < t \leq T} Reg(\pi^T, [t', t], i) \leq \alpha$$

We observe that adaptive regret is a special case of subsequence regret on the action-independent subsequences $\mathcal{E}_{\text{Adapt}} = \{E_{s',s}\}_{0<s'<s\leq T}$, where $E_{s',s}(t) = 1$ if $s' \leq t \leq s$ and $E_{s',s}(t) = 0$ otherwise. Observe that $|\mathcal{E}_{\text{Adapt}}| \leq T^2$, and so we can apply Theorem 16 to immediately conclude:

Theorem 19 *When instantiated with $\mathcal{E} = \mathcal{E}_{Adapt}$, Algorithm 13 obtains adaptive regret bounded by:*

$$\max_{i \in [k], 0 < t' < t \leq T} Reg(\pi^T, [t', t], i) \leq 4\sqrt{T(2 \ln T + \ln k)}$$

3.3.1.2 Group-wise Regret

Sometimes, we will receive information or *context* $x^t \in \mathcal{X}$ about the decision we are about to make at round t before we make it. For example, when making a weather prediction, we might get to observe atmospheric measurements; when making healthcare decisions we will observe an individual's medical history and current vitals, etc. We might want to obtain diminishing regret to the best fixed action in hindsight not just overall, but also *conditional* on relevant pieces of information. For example, when we are making decisions about people and we have fairness concerns, maybe we want to have no regret overall, but also on subsequences of people corresponding to men and women, and also on subsequences of people corresponding to different ethnicities, and also on subsequences of people corresponding to different income brackets, etc. We might also want to condition on subsequences of people corresponding to individuals that have features that we think are relevant to the task at hand —- in a medical setting, perhaps subsequences of people with high blood pressure, subsequences of people with a family history of diabetes, etc. The key thing is that these subsequences are intersecting: a single person will have a gender, an ethnicity, an income bracket, a unique medical history, etc. So it would not make sense to try and run a different no regret algorithm for people with each of the characteristics that we care about, since what would we do when we encountered someone who fit into more than one group?

We can nevertheless ask for a single algorithm to make decisions that has diminishing regret on *all* of the subsequences defined by the groups we care about.

Definition 31 (Group-Wise Regret) *Fix an action space $\mathcal{A} = [k]$, a context space $\mathcal{X}$, a collection of* groups *$\mathcal{G} \in 2^{\mathcal{X}}$ and a transcript $\pi^T = \{x^t, p^t, c^t\}_{t=1}^T$. The expected regret to action $i \in [k]$ on the subsequence corresponding to group $G \in \mathcal{G}$ is::*

$$Reg(\pi^T, G, i) = \sum_{t:x^t \in G} (\langle p^s, c^s \rangle - c_i^s)$$

The learner has expected groupwise regret over $\mathcal{G}$ bounded by α if:

$$\max_{G \in \mathcal{G}} Reg(\pi^T, G, i) \leq \alpha$$

We observe that groupwise regret is a special case of subsequence regret on the action-independent subsequences $\mathcal{E}_{\mathcal{G}} = \{E_G\}_{G \in \mathcal{G}}$, where $E_G(x^t) = 1$ if $x^t \in G$ and $E_G(x^t) = 0$ otherwise. Observe that $|\mathcal{E}_{\mathcal{G}}| = |\mathcal{G}|$, and so we can apply Theorem 16 to immediately conclude:

Theorem 20 *Fix any collection of groups $\mathcal{G} \in 2^{\mathcal{X}}$. When instantiated with $\mathcal{E} = \mathcal{E}_{\mathcal{G}}$, Algorithm 13 obtains adaptive regret bounded by:*

$$\max_{G \in \mathcal{G}} Reg(\pi^T, G, i) \leq 4\sqrt{T(\ln |\mathcal{G}| + \ln k)}$$

3.3.2 General Subsequences

The general form of Algorithm 13 was to weight each action with probability proportional to the (exponential of) the regret to that action on each subsequence, weighted by the current activation of each of the subsequences. This worked when the subsequences had activations $E(t, x^t)$ that were independent of the action a^t chosen at round t — but in general, does not make sense when the events $E(t, x^t, a^t)$ depend on the action chosen. The solution (since we need to select an action from a distribution that depends in some way on the subsequence activations, which in turn depend on which action we pick) is to compute a distribution by solving for a fixed point. This is what the general subsequence regret algorithm below (Algorithm 14) does. The algorithm will seem mysterious at first blush, but we will derive it from first principles in the proof of Theorem 21.

Algorithm 14 Getting General Diminishing Subsequence Regret

Given A collection $\mathcal{E}$ of m subsequence selection functions.
for $t = 1$ to T **do**
Observe context x^t
For each $E \in \mathcal{E}$ and each $i \in [k]$, define the weights:

$$w_{E,i}^t = \frac{\exp\left(\frac{\eta}{2}\sum_{t'=1}^{t-1}\mathbb{E}_{j'\sim p^{t'}}\left[E(t',x^{t'},j')\left(c_{j'}^{t'}-c_i^{t'}\right)\right]\right)}{\sum_{E',j}\exp\left(\frac{\eta}{2}\sum_{t'=1}^{t-1}\mathbb{E}_{j'\sim p^{t'}}\left[E(t',x^{t'},j')\left(c_{j'}^{t'}-c_j^{t'}\right)\right]\right)}$$

Define the $k \times k$ matrix A so that:

$$A_{i,j} = \frac{\sum_{E\in\mathcal{E}} w_{E,j}^t E(t,x^t,i)}{\sum_{i'\in[k]}\sum_{E\in\mathcal{E}} w_{E,i'}^t E(t,x^t,j)}$$

Compute $p^t \in \Delta[k]$, a probability distribution such that:

$$Ap^t = p^t$$

i.e. an eigenvector of A with eigenvalue 1.
Play the distribution p^t.

Theorem 21 *Algorithm 14 is well defined (i.e. the claimed eigenvector exists). Moreover, for any collection $\mathcal{E}$ of m subsequence selection functions, Algorithm 13 implements Algorithm 12 for the Subsequence Regret Multiobjective Optimization Game (Definition 29), and hence obtains the regret bound from Theorem 16:*

$$\max_{E\in\mathcal{E},i\in[k]} Reg(\pi^T,E,i) \le 4\sqrt{T(\ln m + \ln k)}$$

Proof 28 *We will derive the algorithm from first principles, which will help build intuition, rather than just proving that it is correct "out of nowhere".*

Algorithm 12 needs to play a distribution p^t at each round that is a minimax equilibrium for the game with utility function defined as:

$$\begin{aligned} u^t(p,c) &= \sum_{E\in\mathcal{E},i\in[k]} w_{E,i}^t \ell_{E,i}^t(p,c) \\ &= \sum_{E\in\mathcal{E},i\in[k]} w_{E,i}^t \mathop{\mathbb{E}}_{j\sim p}\left[E(t,x^t,j)\left(c_j - c_i\right)\right] \end{aligned}$$

where

$$w_{E,i}^t = \frac{\exp\left(\frac{\eta}{2}\sum_{t'=1}^{t-1}\ell_{E,i}^{t'}(p^{t'},c^{t'})\right)}{\sum_{E',j}\exp\left(\frac{\eta}{2}\sum_{t'=1}^{t-1}\ell_{E',j}^{t'}(p^{t'},c^{t'})\right)}$$

We can expand out the expectation and inner product in the utility function:

$$\begin{aligned}
u^t(p,c) &= \sum_{E\in\mathcal{E},i\in[k]} w^t_{E,i} \mathop{\mathbb{E}}_{j\sim p}\left[E(t,x^t,j)\,(c_j-c_i)\right] \\
&= \sum_{i\in[k]}\sum_{j\in[k]}\sum_{E\in\mathcal{E}} w^t_{E,i}p_j E(t,x^t,j)(c_j-c_i) \\
&= \sum_{i\in[k]}\sum_{j\in[k]}\sum_{E\in\mathcal{E}} w^t_{E,i}E(t,x^t,j)p_jc_j - \sum_{i\in[k]}\sum_{j\in[k]}\sum_{E\in\mathcal{E}} w^t_{E,j}E(t,x^t,i)p_ic_j \\
&= \sum_{j\in[k]} c_j \underbrace{\left(\sum_{i\in[k]}\sum_{E\in\mathcal{E}} w^t_{E,i}E(t,x^t,j)p_j - \sum_{i\in[k]}\sum_{E\in\mathcal{E}} w^t_{E,j}E(t,x^t,i)p_i\right)}_{\chi^t_j(p)}
\end{aligned}$$

So:

$$u^t(p,c) = \sum_{j\in[k]} c_j\chi^t_j(p)$$

We know that the value of this game is 0*, and so if* p^* *is a minimax strategy, it must be that for* all $c\in[0,1]^k$*,* $u^t(p^*,c)=\sum_{j\in[k]} c_j\chi^t_j(p^*)\le 0$ *In order for this to be true, it must be that for all* j*,* $\chi^t_j(p^*)\le 0$ *(since otherwise if there was such a* j *such that* $\chi^t_j(p^*)>0$ *the adversary could set* $c_j=1$ *and* $c_{j'}=0$ *for all* $j'\neq j$*, which would lead to* $u^t(p^*,c)>0$*, a contradiction). Moreover, we can see by symmetry that for all* p*,* $\sum_{j=1}^k \chi^t_j(p)=0$*. Therefore all of the inequalities must be equalities: for all* j*,* $\chi^t_j(p)=0$*. From this we get a set of equalities characterizing* p^**: For all* j*:*

$$p^*_j = \frac{\sum_{i\in[k]}\sum_{E\in\mathcal{E}} w^t_{E,j}E(t,x^t,i)p^*_i}{\sum_{i\in[k]}\sum_{E\in\mathcal{E}} w^t_{E,i}E(t,x^t,j)}$$

Note that if we define the $k\times k$ *matrix* A *such that*

$$A_{i,j} = \frac{\sum_{E\in\mathcal{E}} w^t_{E,j}E(t,x^t,i)}{\sum_{i'\in[k]}\sum_{E\in\mathcal{E}} w^t_{E,i'}E(t,x^t,j)}$$

Then these constraints simplify to:

$$Ap^* = p^*$$

In other words, p *is an eigenvector of* A *with eigenvalue* 1*. Observe that since we know the game has value* 0*, we know that such a* p^* *must exist (i.e. this matrix must have an eigenvector with eigenvalue* 1*), and* p^* *must therefore be a minimax strategy.*

3.3.2.1 Swap Regret

Recall that one of the motivating problems for this section was finding a learning dynamic that converges to correlated equilibrium in general games. In Section 3.1, we established that the empirical distribution over a sequence of action profiles would form an ϵ-approximate correlated equilibrium if and only if each player had sufficiently small *swap regret*. i.e. if for each player i:

$$\max_{\phi:\mathcal{A}_i\to\mathcal{A}_i}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{a^t\sim P^t}\left[\left(c_i(a_i^t,a_{-i}^t)-c_i(\phi(a_i^t),a_{-i}^t\right)\right]\leq\alpha T$$

We first observe that in a k action game, swap regret is at most k times larger than the *subsequence regret* over the set of k subsequences $\mathcal{E}=\{E_a\}_{a\in[k]}$ corresponding to rounds in which the learner i played a particular action a: $E_a(a_i^t)=\mathbb{1}[a_i^t=a]$. This kind of subsequence regret is also called *internal regret*.

Definition 32 *A learner has internal regret bounded by α if they have subsequence regret bounded by α for the collection of subsequences $\mathcal{E}_{Int}=\{E_a\}_{a\in[k]}$, where for each a, $E_a(a^t)=\mathbb{1}[a^t=a]$. In other words, if for each $i\in[k]$ and for each $a\in[k]$:*

$$Reg(\pi^T,E_a,i)=\sum_{t=1}^{T}p_a^t\left(c_a^t-c_i^t\right)\leq\alpha$$

Lemma 3.3.1 *If a k-action learner has internal regret bounded by α on transcript π^T, then their swap regret is bounded by $k\alpha$ — i.e. for every $\phi:\mathcal{A}\to\mathcal{A}$:*

$$\sum_{t=1}^{T}\mathop{\mathbb{E}}_{a^t\sim p^t}\left[\left(c_{a^t}^t-c_{\phi(a^t)}^t\right)\right]\leq k\alpha$$

Proof 29 *We calculate:*

$$\begin{aligned}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{a^t\sim p^t}\left[\left(c_{a^t}^t-c_{\phi(a^t)}^t\right)\right] &= \sum_{t=1}^{T}\sum_{a\in\mathcal{A}}p_a^t\cdot\left(c_a^t-c_{\phi(a)}^t\right)\\ &= \sum_{a\in\mathcal{A}}\sum_{t=1}^{T}p_a^t\cdot\left(c_a^t-c_{\phi(a)}^t\right)\\ &= \sum_{a\in\mathcal{A}}Reg(\pi^T,E_a,\phi(a))\\ &\leq k\alpha\end{aligned}$$

We can therefore invoke Theorem 21 to show that Algorithm 14 can be used to obtain low internal (and hence swap) regret:

Theorem 22 *When Algorithm 14 is instantiated with the k events $\mathcal{E}_{Int}$, it guarantees internal regret bounded by:*

$$\max_{a\in[k],i\in[k]} Reg(\pi^T, E_a, i) \leq 4\sqrt{T(2\ln k)}$$

By Lemma 3.3.1 it therefore also guarantees swap regret bounded by:

$$\max_{\phi:\mathcal{A}\to\mathcal{A}} \sum_{t=1}^{T} \mathop{\mathbb{E}}_{a^t\sim p^t}\left[\left(c^t_{a^t} - c^t_{\phi(a^t)}\right)\right] \leq 4k\sqrt{T(2\ln k)}$$

Recall that Lemma 3.1.1 tells us that if all n players in a general-sum game have swap regret bounded by α over a T-round interaction, then the uniform distribution over their empirical history of play forms an ϵ-correlated equilibrium for $\epsilon = \frac{\alpha}{T}$. Therefore, plugging in $\alpha = 4k\sqrt{T(2\ln k)}$ and solving for T, we find that:

Corollary 3.3.1 *If all players in a k-action general sum game choose their actions over a sequence of T rounds using Algorithm 14 instantiated with events $\mathcal{E}_{int}$, then their empirical history of play forms an ϵ-approximate correlated equilibrium so long as:*

$$T \geq \frac{32k^2\ln(k)}{\epsilon^2}$$

3.3.2.2 Mixing and Matching Guarantees

Our multi-objective/subsequence regret framework gives us the flexibility to easily mix and match guarantees. For example, given k actions $\mathcal{A}$ and a collection of groups $\mathcal{G} \in 2^{\mathcal{X}}$ if we instantiate Algorithm 14 with the events $\mathcal{E} = \mathcal{E}_{\mathcal{G}} \times \mathcal{E}_{\text{int}} = \{E_G(x_t)\cdot E_a(a^t)\}_{G\in\mathcal{G},a\in\mathcal{A}}$, then we get an algorithm that has *groupwise* swap regret. Similarly if we instantiate the algorithm with the events $\mathcal{E}_{\text{Adapt}} \times \mathcal{E}_{\text{int}}$ we get adaptive-swap regret. We can similarly get groupwise adaptive swap regret, and a variety of other kinds of guarantees.

Bibliographic Notes and Further Reading

See Williams [1980] for a sufficient condition for the existence of Nash equilibria in continuous general sum games. The multi-objective optimization framework from Section 3.2 is from Lee et al. [2022], who derive the algorithm from first principles (deriving exponential weights along the way). Haghtalab et al. [2023b] give a variant of this reduction that modularly derives multiobjective guarantees from an arbitrary no-regret algorithm, which is the exposition we follow here. Swap regret was first studied by Foster and Vohra [1999]. Blum and Mansour [2007] gave efficient algorithms for swap and subsequence regret that are very similar to the ones we derive here. Algorithms for adaptive regret have been given by Hazan and Seshadhri [2009], Adamskiy et al. [2012]. Blum and Lykouris [2020] gave the first algorithms for obtaining diminishing groupwise regret.

4

Making Unbiased Predictions and Calibration

CONTENTS

Thus far we have studied the design of decision making algorithms: i.e. algorithms whose output is a *choice of which action to take* — that promise various kinds of regret guarantees. But suppose instead we want to provide a forecast for some outcome that *other* (diverse) decision makers can use to make decisions downstream, depending on their own utility functions? This will bring us to the topic of calibrated forecasting — as well as a variety of related guarantees.

To start, we will ignore the incentives of the downstream decision maker, and ask whether we can make predictions that will guarantee that downstream decision makers will have no (external) regret if they act as if our predictions are correct. Next, we'll ask whether we can make predictions that will offer stronger guarantees to the downstream decision makers, and incentivize them to follow our predictions.

To answer this question, we'll will start out by looking at a model in which we aim to predict, in sequence, an adversarially chosen vector valued outcome such that our predictions are *unbiased* — i.e. such that on average over the sequence, the average of our predictions will be equal to the average of the realized outcomes. This will turn out to be too easy of a goal: we can obtain

it simply by predicting what happens today is what happened yesterday, and correspondingly, such *marginally* unbiased predictions will not be very useful. But we can use the multiobjective framework we developed in Chapter 3 to ask for more: unbiasedness not just marginally, but simultaneously over many different subsequences, which can be defined in terms of our predictions. As we will see, if we choose these subsequences correctly, our predictions *will* become useful for downstream decision makers.

4.1 Modeling Decision Makers

A decision maker is defined by an action space and a utility function. We will assume in this Chapter that the decision maker's utility function depends on the action they take and on some vector-valued *state* that is unknown at the time that they must make their prediction. We will assume that their utility is linear in this state vector.

Definition 33 *A decision maker is defined by a collection of actions $\mathcal{A}$, a convex state space $\mathcal{C} \subset \mathbb{R}^d$, and a utility function $u : \mathcal{A} \times \mathcal{C} \to [0, 1]$ such that for every $a \in \mathcal{A}$, $u(a, \cdot)$ is a linear function of the state s. We will also assume that $u(a, \cdot)$ is L-Lipschitz (in the L_∞-norm) in its second argument, for all a. That is for all $a \in \mathcal{A}$, and for all $s, s' \in \mathbb{R}^d$: $|u(a, s) - u(a, s')| \leq L \max_{i \in [d]} |s_i - s'_i|$.*

Remark 4.1.1 *The assumption that u is linear in the state s is not very restrictive. First note that it easily generalizes to utility functions that are* affine *in the state, by simply adding an extra dimension to the state space that always takes value* 1*. This preserves the convexity of $\mathcal{C}$, and now allows a constant offset in the utility for each action. If the dimension of the state is equal to the number of actions $d = |\mathcal{A}|$, then we can encode arbitrary payoffs for each of the d actions in coordinate s_i. In this case, $u(i, s) = \langle e_i, s \rangle$, which is linear in s, and places no restrictions at all on the payoff the agent gets from each action. Similarly, suppose for each action a, the decision maker has arbitrary utility for each of d possible outcomes that might be realized. Here s can represent a probability distribution over these d outcomes: for a risk neutral decision maker (who cares about maximizing their expected utility), their (expected) utility is a linear function of s by linearity of expectation. Both of these examples result in a Lipschitz utility function as well. We will see more examples.*

Suppose that in a sequential decision making setting, we make predictions $\hat{s}_t \in \mathcal{C}$ at each round t of the unknown state s_t The decision maker can choose what action to take at each round using a *policy* that maps a prediction to an action. We will pay particular attention to the *best response policy*, which for a decision maker with utility function u, in response to a prediction $\hat{s}_t$, plays

the action that would maximize utility if the true state was actually realized to be $s_t = \hat{s}_t$.

Definition 34 *Fixing a prediction space $\mathcal{C}$ and an action space $\mathcal{C}$, A policy is a function $P : \mathcal{C} \to \mathcal{A}$. For a utility function u, the* best response policy *for u is:*

$$BR(u, \hat{s}) = \arg\max_{a \in \mathcal{A}} u(a, \hat{s})$$

Where ties may be broken using any consistent rule.

Remark 4.1.2 *A decision maker could use information* other *than the prediction $\hat{s}$ to take an action, but we will restrict attention here to decision makers who have no other source of information about the state and so use some policy as we have defined it.*

Fixing a sequence of predictions $\hat{s}_t$, outcomes s_t, and a policy P for a decision maker induces a sequence of actions for them, which in turn realize utility and regret.

Definition 35 *Fix a sequence of predictions $\hat{s}^t$ and outcomes s^t, as well as a policy P for a decision maker with utility function u and action space $\mathcal{A}$. Their induced regret to an action $a \in \mathcal{A}$ is:*

$$\sum_{t=1}^{T} \big(u(a, s^t) - u(P(\hat{s}^t), s^t)\big)$$

4.2 Predicting for No-Regret Play

There is a simple strategy for making predictions that will guarantee that for any agent with utility function u, if they play their best response policy, they will have no regret to any of their actions. The idea is straightforward: we simply predict the average state that has been observed so far, perturbed with a bit of noise. When downstream agents play the best response policy, they will be implementing a version of "Follow the Perturbed Leader", which we proved in Section 1.4 gives them a no-regret guarantee. In the version we give below, the "predictions" won't quite look like states in $\mathcal{C}$, because of the noise. This isn't hard to fix, and we won't bother for the sake of simplicity. (We'll in any case see how to make predictions with stronger guarantees that avoid this issue shortly).

Algorithm 15 State-Perturbed Follow the Pertured Leader (SP-FTPL): Getting External Regret Guarantees for Downstream Actors

for $t = 1$ to T **do**

Predict:

$$\hat{s}^t = \frac{1}{t-1}\left(\sum_{t'=1}^{t-1} s^{t'} + N^t\right)$$

where $N^t \sim U[0, 1/\epsilon]^d$ is a uniformly random noise vector.

Any decision maker who follows the best response policy for predictions produced by Algorithm 15 will get a no-regret guarantee at the same rate they would had they played Follow the Perturbed Leader — essentially because they will be playing follow the perturbed leader! The only difference is that we are perturbing the signal rather than their utilities directly (which we do not know), but as we will see the effect will be the same. This algorithm has the advantage that it gets no regret guarantees for all downstream decision makers simultaniously.

Theorem 23 *Fix any utility function $u : \mathcal{A} \times \mathbb{R}^d$ that is linear and L-Lipschitz in its 2nd argument. Then for any sequence of states $s^1, \ldots, s^T$ a decision maker with utility function u following the best response policy with respect to the forecasts $\hat{s}$ made by Algorithm 15 will have expected regret at most:*

$$\sum_{t=1}^{T}\left(u(a, s^t) - \mathop{\mathbb{E}}_{N^t}[u(BR(u, \hat{s}^t), s^t)]\right) \leq \frac{L}{\epsilon} + 2C\epsilon \cdot T$$

where $C = \max_{s \in \mathcal{C}} ||s||_1$.

Remark 4.2.1 *Choosing $\epsilon = \sqrt{\frac{L}{2CT}}$ to minimize this bound we get a final regret bound of*

$$\sum_{t=1}^{T}\left(u(a, s^t) - \mathop{\mathbb{E}}_{N^t}[u(BR(u, \hat{s}^t), s^t)]\right) \leq 2\sqrt{2CLT}$$

Our analysis will mirror our analysis of Follow the Perturbed Leader from Section 1.4, by going through two intermediate algorithms. State Be the Leader (SBL) will correspond to the algorithm that at every round predicts state $\bar{s}^t = \frac{1}{t}\sum_{t'=1}^{t} s^{t'}$ (i.e. the average of the states up to and including round t), and State-Perturbed-Be-the-Perturbed-Leader (SP-BTPL) will be the algorithm that predicts state $\tilde{s}^t = \frac{1}{t}\left(\sum_{t'=1}^{t} s^{t'} + \tilde{N}^t\right)$ where $\tilde{N}^t \sim U[0, 1/\epsilon]^d$

We first observe that for any utility function u that is linear in the state space, following the predictions of SBL will result in no regret.

Lemma 4.2.1 *Fix any utility function $u : \mathcal{A} \times \mathbb{R}^d$ that is linear in its second*

argument. Then for any sequence of states $s^1, \ldots, s^T$, a decision maker with utility function u following the best response policy with respect to $\bar{s}^t$ will have no external regret. In other words for all $a \in \mathcal{A}$:

$$\sum_{t=1}^{T} \left(u(a, s^t) - u(BR(u, \bar{s}^t), s^t)\right) \leq 0$$

Proof 30 *We will establish that if the decision maker is best responding to predictions $\bar{s}^t$, then they are in fact taking the same sequence of actions that they would have taken had they been directly implementing "Be the Leader". That is, we will show that:*

$$BR(u, \bar{s}^t) \in \arg\max_{a \in \mathcal{A}} \frac{1}{T} \sum_{t=1}^{T} u(a, s^t)$$

Once we establish this, the claim follows from Lemma 1.4.1 which bounds the regret of Be the Leader. This equivalence follows because of the linearity of u in the state. For every action $a \in \mathcal{A}$:

$$\begin{aligned} u(a, \bar{s}^t) &= u\left(a, \frac{1}{t} \sum_{t'=1}^{t} s^{t'}\right) \\ &= \frac{1}{T} \sum_{t'=1}^{t} u(a, s^{t'}) \end{aligned}$$

Thus

$$BR(u, \bar{s}^t) \in \arg\max_{a \in \mathcal{A}} \frac{1}{T} \sum_{t=1}^{T} u(a, s^t)$$

which is what we needed to establish.

Next, we establish that following the predictions of SP-BTPL still leads to low regret — informally, because the noise cannot affect the agent's utility substantially, because her utility function is Lipschitz.

Lemma 4.2.2 *Fix any utility function $u : \mathcal{A} \times \mathbb{R}^d$ that is linear and L-Lipschitz in its second argument. Then for any sequence of states $s^1, \ldots, s^T$, a decision maker with utility function u following the best response policy with respect to $\tilde{s}^t$ will have expected external regret bounded by:*

$$\max_{a \in \mathcal{A}} \left(\sum_{t=1}^{T} u(a, s^t) - \mathop{\mathbb{E}}_{\tilde{N}^1, \ldots, \tilde{N}^T} \left[\sum_{t=1}^{T} u(a, BR(u, \tilde{s}^t), s^t) \right] \right) \leq \frac{L}{\epsilon}$$

Proof 31 *Just as with our analysis of follow the perturbed leader, we will reduce our analysis of SP-BTPL to the analysis we just completed of SBL, on an alternative sequence of states that includes the noise added by the algorithm.*

First we note that to bound expected regret we can imagine that the algorithm uses the same noise at every round. To make the dependence explicit, we write $\tilde{s}^t(N^t)$ *to reflect the dependence on the forecast that SP-BTPL makes at round* t *on the noise* N^t*:*

$$\begin{aligned}\mathop{\mathbb{E}}_{\tilde{N}^1,\ldots,\tilde{N}^T}\left[\sum_{t=1}^{T} u(a, BR(u, \tilde{s}^t(\tilde{N}^t)), s^t)\right] &= \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\tilde{N}^t}[u(a, BR(u, \tilde{s}^t(\tilde{N}^t)), s^t)] \\ &= \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\tilde{N}}[u(a, BR(u, \tilde{s}^t(\tilde{N})), s^t)]\end{aligned}$$

where $N \sim U[0, 1/\epsilon]^d$ *is a single noise vector distributed uniformly at random in each coordinate. Thus in the remainder we analyze a sequence of forecasts* $\tilde{s}^1(\tilde{N}), \ldots, \tilde{s}^T(\tilde{N})$ *that results from using a variant of SP-BTPL that uses the same random perturbation at each round. Observe that we can interpret* $\tilde{s}^t(\tilde{N})$ *by imagining that the noise is a perturbation of the first state* s^1*:*

$$\tilde{s}^t(\tilde{N}) = \frac{1}{t}\left(\sum_{t'=1}^{t} s^{t'} + \tilde{N}\right) = \frac{1}{t}\left((s^1 + \tilde{N}) + \sum_{t'=2}^{t} s^{t'}\right)$$

Thus we can view SB-BTPL as being equivalent to SBTL on this modified sequence. Applying Lemma 4.2.1 we have that for all $\tilde{N} \in [0, 1/\epsilon]^d$ *and all* $a \in \mathcal{A}$*:*

$$\begin{aligned}\sum_{t=1}^{T} u(a, BR(u, \tilde{s}^t(\tilde{N})), s^t)] &\geq u(a, (s^1 + \tilde{N}) + \sum_{t=2}^{T} u(a, s^t) \\ &\geq \sum_{t=1}^{T} u(a, s^t) - \frac{L}{\epsilon}\end{aligned}$$

where in the last inequality we use the fact that $||\tilde{N}||_\infty \leq \frac{1}{\epsilon}$ *and that* u *is* L*-Lipschitz in its second argument.*

Finally (again, just as in our analysis of FTPL), we observe that at every round, SP-BTPL and SP-FTPL play a very similar distribution over actions and so must have a similar regret bound.

Lemma 4.2.3 *Let* $C = \max_{s \in \mathcal{C}} ||s||_1$*. Then there is a coupling between* N^t *and* $\tilde{N}^t$ *such that with probability at least* $1 - 2C\epsilon$*,* $\hat{s}^t(N^t) = \tilde{s}^t(\tilde{N}^t)$*. A consequence of this is that for any policy* P *mapping forecasts to actions, and any utility function* $u(a, s)$ *taking values in* $[0, 1]$*:*

$$\mathop{\mathbb{E}}_{N^1,\ldots,N^T}\left[\sum_{t=1}^{T} u(P(\hat{s}^t(N^t)), s^t)\right] \geq \mathop{\mathbb{E}}_{\tilde{N}^1,\ldots,\tilde{N}^T}\left[\sum_{t=1}^{T} u(P(\tilde{s}^t(\tilde{N}^t)), s^t)\right] - 2C\epsilon \cdot T$$

The proof of this lemma follows very similar arguments as in Section 1.4 and so we will not repeat it. In all, we have the ingredients to prove Theorem 23:

Proof 32 (Proof of Theorem 23) *Combining the statements of Lemmas 4.2.3 and 4.2.2, we have for every action $a \in \mathcal{A}$:*

$$\begin{aligned}\mathbb{E}\left[\sum_{t=1}^{T} u(BR(u,\hat{s}^t), s^t)\right] &\geq \mathbb{E}\left[\sum_{t=1}^{T} u(BR(u,\tilde{s}^t), s^t)\right] - 2C\epsilon \cdot T \\ &\geq \sum_{t=1}^{T} u(a, s^t) - \frac{L}{\epsilon} - 2C\epsilon \cdot T\end{aligned}$$

which completes the proof.

Thus we have an algorithm for producing a single set of predictions that can be used by any number of downstream decision-makers with different utility functions. If they treat our predictions as correct (i.e. as if our predicted state is the true state) and best respond to them, then they will all have diminishing (external) regret. Pretty cool! But there are a few reasons why we might want more.

1. Why should downstream agents treat our predictions as correct? Rather than best responding to them, they could use some *other* policy mapping our predictions to actions, and we have not given any reason to think that the *straightforward* policy which plays the best response to the predicted state is the best policy. If downstream agents are rational, they may therefore not best respond to our predictions.
2. Can we offer downstream agents guarantees that are stronger than external regret guarantees? What about the other kinds of regret guarantees we have studied, when we had the luxury of designing the decision maker's algorithm: swap regret, adaptive regret, subsequence regret, groupwise regret, and so on. Can we make predictions such that many different downstream agents with different utility functions will all have these stronger forms of regret if they best respond to our predictions?

Towards answering these questions, we'll define a more general task that will turn out to be useful: making *unbiased* predictions.

4.3 A Model for Unbiased Prediction

In our model for prediction, there will be a d dimensional prediction space and outcome space. In rounds, the predictor will choose a (distribution over) prediction(s) from the prediction space, and then the adversary will choose an outcome. Our eventual goal will be that the predictions of the predictor

should equal the outcomes chosen by the adversary *on average* over various subsequences.

Definition 36 (Adversarial Prediction) *Fix a convex bounded* prediction *space $\mathcal{C} \subset [0,1]^d$ and a context space $\mathcal{X}$. The adversarial prediction game proceeds in rounds $t = 1, \ldots, T$:*

1. The learner observes context $x^t \in \mathcal{X}$, chosen by an adversary.

2. With knowledge of $s^1, \ldots, s^{t-1}$ the learner chooses a distribution over predictions $p^t \in \Delta\mathcal{C}$.

3. With knowledge of $p^1, \ldots, p^t$ the adversary chooses an outcome $s^t \in \mathcal{C}$.

4. The instantaneous expected bias *of the learner's prediction is $b_L^t = \mathbb{E}_{\hat{s}_t \sim p^t}[\hat{s}^t - p^t] = \mathbb{E}_{\hat{s}_t \sim p^t}[\hat{s}^t] - p^t$.*

Given a T round interaction, the realized predictions and outcomes are accumulated in a transcript $\pi^T = \{(x^1, p^1, s^1), \ldots, (x^T, p^T, s^t)\}$

There are various natural decision spaces. For example, perhaps we aim to predict one of k disjoint weather outcomes (e.g. it might be sunny, or cloudy, or rainy, or snowy...), in which case the prediction/outcome space would naturally be $\mathcal{C} = \Delta[k]$, the probability simplex over k outcomes. In this case we would predict a distribution on outcomes ("a 20% chance of rain, a 30% chance of snow, ..."), and what the adversary would realize is a deterministic outcome (e.g. a standard basis vector e_i corresponding to the event that outcome i is realized). Alternately, we might aim to predict the *gains* that we would obtain by taking one of k actions, as in the experts learning problem. In this case, our prediction and outcome space might naturally be $\mathcal{C} = [0,1]^k$, the unit k-dimensional hyper-cube. Or in an online linear optimization/online combinatorial optimization setting like an online shortest paths problem, we might aim to predict the cost or congestion associated with each of d atomic actions. One can imagine other choices as well.

The simplest goal we can start from is that our predictions be unbiased on average, no matter what sequence of outcomes is realized. Our predictions (and therefore our bias) are vector-valued quantities. We will quantify bias using the ℓ_∞ norm, which asks that our maximum bias across all coordinates is bounded.

Definition 37 *Fixing a transcript π^T, the learner's expected bias after T rounds is:*

$$Bias(\pi^T) = \left\| \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t}[\hat{s}^t] - s^t \right\|_\infty = \max_{i \in [d]} \left| \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t}[\hat{s}_i^t] - s_i^t \right|$$

If all we want of our predictions is that they have small bias, then there is a simple algorithm that can obtain it: simply predict that the realization today will be the same as yesterday: $\hat{s}^t = s^{t-1}$.

Algorithm 16 Making Unbiased Predictions

Let $s^0 = 0$.
for $t = 1$ to T **do**
  Observe context x^t (and ignore it).
  Predict $\hat{s}^t = s^{t-1}$, and observe s^t.

Claim 4.3.1 *Algorithm 16 makes predictions such that for any sequence of realized outcomes, the bias of the predictions after T rounds is at most:*

$$Bias(\pi^T) = \left\| \sum_{t=1}^{T} \hat{s}^t - s^t \right\|_\infty \leq 1$$

Proof 33 *Consider any coordinate $i \in [d]$. We have:*

$$\begin{aligned} \left| \sum_{t=1}^{T} (\hat{s}_i^t - s_i^t) \right| &= \left| \sum_{t=1}^{T} (s_i^{t-1} - s_i^t) \right| \\ &= \left| \sum_{t=1}^{T} s_i^t - \sum_{t=1}^{T-1} s_i^t \right| \\ &= \left| s_i^T \right| \\ &\leq 1 \end{aligned}$$

This is a terrific bound — the bias doesn't grow with time at all, and is in fact universally upper bounded by 1. But the algorithm is self-evidently "too simple": how can predicting that tomorrow's outcome will be the same as yesterdays provide any information that is useful for decision making? Indeed it does not (at least not directly). One way to see this, which foreshadows how we will use predictions that satisfy more sophisticated bias guarantees, is to ask whether a decision maker who treats the forecasts as correct will perform well as measured by regret. Consider an expert learning problem with two experts: our goal is to forecast the costs of the two experts—$\mathcal{C} = [0,1]^2$. Suppose the actual sequence of costs is the sequence we saw in Chapter 1 which demonstrated why the "Follow the Leader" algorithm obtains linearly growing regret:

$$c^1 = (1/2, 0) \;\; c^2 = (0,1) \;\; c^3 = (1,0) \;\; c^4 = (0,1) \;\; c^5 = (1,0), \;\; c^6 = (0,1), \ldots$$

In this case, by consistently predicting that yesterday's costs will be repeated today (when in fact they alternate), we will consistently mislead a downstream decision maker about which action is best, leading once again to linearly growing regret.

The solution will be to ask for more than *marginally* unbiased predictions.

4.3.1 Conditionally Unbiased Prediction

To define conditionally unbiased prediction, we introduce *events* similar to those we used in defining subsequence regret in Chapter 3.

Definition 38 *In an adversarial prediction setting, an event is defined by a subsequence selection function $E : [T] \times \mathcal{C} \times \mathcal{X} \to [0, 1]$. Fixing a transcript $\pi^T = \{(x^1, p^1, s^1), \ldots, (x^T, p^T, s^t)\}$, the bias with respect to event E is:*

$$Bias(\pi^T, E) = \left\| \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t} [E(t, \hat{s}^t, x^t) \cdot \left(\hat{s}^t - s^t\right)] \right\|_\infty$$

For a collection of events $\mathcal{E}$, we say that a transcript π^T has $\mathcal{E}$-bias bounded by α if:

$$Bias(\pi^T, \mathcal{E}) = \max_{E \in \mathcal{E}} Bias(\pi^T, E) \leq \alpha$$

As before, when writing subsequence selection functions, we will elide parameters that are not used. For example, if an event E is independent of the round and the context, we will write $E(\hat{s}^t)$ rather than $E(t, \hat{s}^t, x^t)$.

Given a finite collection of events $\mathcal{E}$, we can reduce the problem of making $\mathcal{E}$-unbiased predictions to the problem of multiobjective optimization, using the framework we developed in Section 3.2. To do so, we will define a $2 \cdot d \cdot |\mathcal{E}|$-coordinate multiobjective optimization game, with one coordinate for each dimension d of our prediction space, each event $E \in \mathcal{E}$, and each sign $\sigma \in \{-1, 1\}$ (to account for the fact that we don't want the bias in each coordinate to have large magnitude in either the positive or negative direction). The action space for both the learner and the adversary will consist of the outcome/prediction space $\mathcal{C}$ — but to satisfy the requisite technical condition that the loss functions be convex in the learner's actions, we will need to allow the learner to randomize over a finite number of discrete points in their prediction space. The formal construction follows.

Definition 39 *$\mathcal{C}_\epsilon \subset \mathcal{C} \subset \mathbb{R}^d$ is an ℓ_∞ ϵ-net for a set $\mathcal{C}$ if for every $s \in \mathcal{C}$, there exists an $s' \in \mathcal{C}_\epsilon$ such that $||s - s'||_\infty \leq \epsilon$.*

Observation 4.3.1 *For any $\mathcal{C} \subseteq [0, 1]^d$, simply by discretizing each coordinate of vectors $s \in \mathcal{C}$ to multiples of ϵ, we obtain a finite ϵ-net $\mathcal{C}_\epsilon$ of size $|\mathcal{C}_\epsilon| \leq \left(\frac{1}{\epsilon}\right)^d$.*

Definition 40 (ϵ-Encoding $\mathcal{E}$-unbiased Prediction as Multiobjective Optimization) *Given a d-dimensional instance of the adversarial prediction problem and a set of events $\mathcal{E}$, we construct a $2d|\mathcal{E}|$-dimensional instance of the multiobjective optimization game as follows: At each round t:*

> *1. The strategy space for the Learner is $\mathcal{A}^t = \Delta\mathcal{C}_\epsilon$, the set of distributions over some finite ϵ-net of $\mathcal{C}$.*

2. The strategy space for the Adversary is $\mathcal{B}^t = \mathcal{C}$.

3. For each $\sigma \in \{-1, 1\}$, $E \in \mathcal{E}$, *and* $i \in [d]$ *we construct a loss increment function:*

$$\ell_{\sigma,E,i}(p^t, s^t) = \sigma \cdot \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t}[E(t, \hat{s}^t, x^t)(\hat{s}_i^t - s_i^t)]$$

First, we verify that our encoding in Definition 40 satisfies the requirements of a multiobjective optimization game, and bound the adversary-moves-first value of the game.

Lemma 4.3.1 *The encoding in Definition 40 satisfies the requirements of a multiobjective optimization game and at each round has adversary moves first value bounded by:*

$$v_A^t = \max_{s \in \mathcal{B}^t} \min_{p \in \mathcal{A}^t} \max_{\sigma,\mathcal{E},i} \ell_{\sigma,E,i}^t(p, s) \leq \epsilon$$

Proof 34 *By assumption,* $\mathcal{B}^t = \mathcal{C}$ *is a closed* d*-dimensional convex set. Similarly, since* $\mathcal{C}_\epsilon$ *is finite,* $\Delta\mathcal{C}_\epsilon$ *is a finite dimensional closed convex set. We now consider the loss increments:*

$$\ell_{\sigma,E,i}(p^t, s^t) = \sigma \cdot \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t}[E(t, \hat{s}^t, x^t)(\hat{s}_i^t - s_i^t)]$$

These functions are linear (and hence concave) in the adversary's action s^t*, and by linearity of expectation, are linear (and hence convex) in the learner's action* p^t*; thus we have established that the encoding satisfies the requirements of a multiobjective optimization game. Next to bound the adversary moves first value of the game.*

Since $\mathcal{B}^t = \mathcal{C}$ *and* $\mathcal{A}^t = \Delta\mathcal{C}_\epsilon$ *(a distribution over an* ϵ*-net of* $\mathcal{B}^t$*), for any* $s \in \mathcal{B}^t$*, the Learner can best respond with a distribution* $p(s) \in \mathcal{A}^t$ *that places all of its weight on* $s' \in \mathcal{A}^t$ *such that* $||s - s'||_\infty \leq \epsilon$*. Therefore:*

$$\begin{aligned} v_A^t &= \max_{s \in \mathcal{B}^t} \min_{p \in \mathcal{A}^t} \max_{\sigma,\mathcal{E},i} \ell_{\sigma,E,i}^t(p, s) \\ &\leq \max_{s \in \mathcal{B}^t} \max_{\sigma,\mathcal{E},i} \ell_{\sigma,E,i}^t(p(s), s) \\ &= \sigma E(t, s', x^t)(s'_i - s_i) \\ &\leq \epsilon \end{aligned}$$

Next, we verify that our encoding is correct: namely that a bound on the adversary-moves-first regret in the multiobjective optimization game defined in Definition 40 in fact corresponds to a bound on the $\mathcal{E}$-bias in the original adversarial prediction game.

Lemma 4.3.2 *Fix an adversarial prediction game, and consider the corresponding* ϵ*-encoding as a multiobjective optimization game defined in Definition 40. Then a transcript* $\pi^T = \{(x^1, p^1, s^1) \ldots, (x^T, p^T, s^T)\}$ *that has Adversary-Moves-First regret bounded by* α *also has* $\mathcal{E}$*-bias bounded by* $\epsilon \cdot T + \alpha$*:*

$$Bias(\pi^T, \mathcal{E}) \leq \epsilon \cdot T + Reg_{AMF}(\pi^T)$$

Proof 35 *We can calculate:*

$$\begin{aligned}
Reg_{AMF}(\pi^T) &= \max_{\sigma,E,i} \sum_{t=1}^{T} (\ell^t_{\sigma,E,i}(p^t, s^t) - v^t_A) \\
&\geq \max_{\sigma,E,i} \sum_{t=1}^{T} \ell^t_{\sigma,E,i}(p^t, s^t) - \epsilon T \\
&= \max_{\sigma,E,i} \sum_{t=1}^{T} \sigma \cdot \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t} [E(t, \hat{s}^t, x^t)(\hat{s}^t_i - s^t_i)] - \epsilon T \\
&= \max_{E} \left\| \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t} [E(t, \hat{s}^t, x^t)(\hat{s}^t - s^t)] \right\|_\infty - \epsilon T \\
&= Bias(\pi^T, \mathcal{E}) - \epsilon T
\end{aligned}$$

Rearranging terms gives the lemma.

Therefore, we can apply Theorem 15 to conclude that using Algorithm 12 to make predictions guarantees bounded conditional bias, for any finite set of conditioning events $\mathcal{E}$:

Theorem 24 *Fix an adversarial prediction game with a set of events $\mathcal{E}$. Using the ϵ-encoding as a multiobjective optimization game defined in Definition 40, Algorithm 12 guarantees that for any sequence of realizations $s^1, \ldots, s^T$, the $\mathcal{E}$-bias on the resulting transcript is bounded by:*

$$Bias(\pi^T, \mathcal{E}) \leq 4\sqrt{T \ln(2 \cdot |\mathcal{E}|d)} + \epsilon T$$

Proof 36 *This follows from applying Lemma 4.3.2 and the AMF regret bound given in Theorem 15 to our multiobjective game, which has $2d|\mathcal{E}|$ objective coordinates.*

Remark 4.3.1 *If we instantiate Theorem 24 with $\epsilon \leq \frac{1}{\sqrt{T}}$, then the bias we obtain is at most $5\sqrt{T \ln(2 \cdot |\mathcal{E}|d)}$.*

Observe that it is not clear how we can efficiently *run* Algorithm 12, which in our case requires computing a minimax equilibrium at each round in a game that has *exponentially many* (in d) actions for the learner, and continuously many d-dimensional actions for the adversary: we will return to this question in Section 4.5. In the mean time we will begin investigating whether there are collections of events $\mathcal{E}$ that make $\mathcal{E}$-unbiased predictions useful for downstream decision makers.

4.4 Calibration

Marginally unbiased predictions (i.e. $\mathcal{E}$-unbiased predictions for $\mathcal{E}$ containing only the constant function $E(t) = 1$) are one extreme of the $\mathcal{E}$-bias spectrum: they ask for very little, and they are very easy to obtain. Calibration will be the other extreme: it will informally ask for unbiasedness of our predictions *conditional on our predictions themselves* (Although for technical reasons we'll need to in fact condition on a small ball of predictions). On the one hand, as we will see, this will offer an extremely strong guarantee to downstream decision makers. On the other hand, it means that the number of events $\mathcal{E}$ we care about will grow exponentially in the dimension d of our predictions, which will make it infeasible to obtain except in very low dimensional prediction settings.

Definition 41 (An m Bucketing) *Let $\mathcal{C}_{1/m}$ be a minimal $(1/m)$-net (in the ℓ_∞ norm) of $\mathcal{C} \subseteq [0,1]^d$. For any $s \in \mathcal{C}$ and $s' \in \mathcal{C}_{1/m}$, say that $s \in B(s')$ if $||s' - s||_\infty \leq ||s'' - s||_\infty$ for all $s'' \in \mathcal{C}_{1/m}$. We break ties arbitrarily so that each $s \in B(s')$ for only a single $s' \in \mathcal{C}_{1/m}$.*

Note we can always have $|\mathcal{C}_{1/m}| \leq m^d$.

Definition 42 *Fix an adversarial prediction setting with outcome space $\mathcal{C} \subseteq [0,1]^d$. We say that a transcript π^T corresponds to (α, m)-calibrated predictions if it has $\mathcal{E}_{m-Cal}$-bias bounded by α for the set of events:*

$$\mathcal{E}_{m-Cal} = \{E_{s'}(\hat{s}^t) = \mathbb{1}[s^t \in B(s')] : s' \in \mathcal{C}_{1/m}\}$$

Calibration will be an attractive guarantee because it will turn out not only that downstream decision makers who best respond to calibrated predictions will have strong regret guarantees, but that it is in fact an optimal policy (amongst all policies mapping predictions to actions) to treat calibrated predictions as correct and best respond to them. To simplify the analysis, we will not release the predictions $\hat{s}^t$ directly to the decision maker, but will first "snap" $\hat{s}^t$ to its nearest point $\tilde{s}^t \in \mathcal{C}_{1/m}$ and release that. Here $\hat{s}^t \in B(\tilde{s}^t)$, and in particular $||\hat{s}^t - \tilde{s}^t|| \leq 1/m$.

Theorem 25 *Fix any utility function $u : \mathcal{A} \times \mathcal{C}$ such that for each action $a \in \mathcal{A}$ $u(a, \cdot)$ is linear and L-Lipschitz in its second argument. Let $P : \mathcal{C} \to \mathcal{A}$ be any policy mapping "snapped" predictions $\tilde{s}$ to actions $\mathcal{A}$. Then for any transcript π^T whose predictions are (α, m)-calibrated:*

$$\sum_{t=1}^{T} \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t} \left[u(BR(u, \tilde{s}^t), s^t)\right] \geq \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t} \left[u(P(\tilde{s}^t), s^t)\right] - 2\alpha \cdot m^d \cdot L - \frac{2T}{m}$$

Proof 37 *We can calculate:*

$$\begin{aligned}
& \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t} \left[u(BR(u, \tilde{s}^t), s^t) \right] \\
= & \sum_{t=1}^{T} \sum_{s' \in \mathcal{C}_{1/m}} \Pr[\hat{s}^t \in B(s')] u(BR(u, s'), s^t) \\
= & \sum_{s' \in \mathcal{C}_{1/m}} u\left(BR(u, s'), \sum_{t=1}^{T} \mathbb{E}[\mathbb{1}[\hat{s}^t \in B(s')]] s^t \right) \\
\geq & \sum_{s' \in \mathcal{C}_{1/m}} \left(u\left(BR(u, s'), \sum_{t=1}^{T} \mathbb{E}[\mathbb{1}[\hat{s}^t \in B(s')]] \hat{s}^t \right) - \alpha L \right) \\
\geq & \sum_{s' \in \mathcal{C}_{1/m}} \left(u\left(BR(u, s'), \sum_{t=1}^{T} \mathbb{E}[\mathbb{1}[\hat{s}^t \in B(s')]] s' \right) - \alpha L - \frac{\sum_{t=1}^{T} \Pr[\hat{s}^t \in B(s')]}{m} \right) \\
= & \sum_{s' \in \mathcal{C}_{1/m}} u\left(BR(u, s'), \sum_{t=1}^{T} \mathbb{E}[\mathbb{1}[\hat{s}^t \in B(s')]] s' \right) - \alpha |\mathcal{C}_{1/m}| L - \frac{T}{m} \\
\geq & \sum_{s' \in \mathcal{C}_{1/m}} u\left(BR(u, s'), \sum_{t=1}^{T} \mathbb{E}[\mathbb{1}[\hat{s}^t \in B(s')]] s' \right) - \alpha \cdot m^d \cdot L - \frac{T}{m} \\
\geq & \sum_{s' \in \mathcal{C}_{1/m}} u\left(P(s'), \sum_{t=1}^{T} \mathbb{E}[\mathbb{1}[\hat{s}^t \in B(s')]] s' \right) - \alpha \cdot m^d \cdot L - \frac{T}{m} \\
\geq & \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t} \left[u(P(\tilde{s}^t), s^t) \right] - 2\alpha \cdot m^d \cdot L - \frac{2T}{m}
\end{aligned}$$

Here we have used the bias condition and the fact that $||\hat{s} - \tilde{s}||_\infty \leq 1/m$ *to compare the expression* $u(BR(u, \tilde{s}^t)), s^t)$ *to* $u(BR(u, \tilde{s}^t), \tilde{s}^t)$*. The best response policy is by definition the optimal policy in this case (and hence obtains higher utility than* P*):*

$$u(BR(u, \tilde{s}^t), \tilde{s}^t) \geq u(P(\tilde{s}^t), \tilde{s}^t)$$

We then use the bias conditions to again compare this to $u(P(\tilde{s}^t), s^t)$*.*

If our goal is to make predictions that downstream agents will be strongly incentivized to treat as correct (and best respond to), then we get to pick the parameter m. Once we do this, we can apply Theorem 24 to find the value of α that our multiobjective optimization algorithm will be able to obtain. In particular, given a choice of m, the number of events that we need to control the bias of is at most m^d, and so we have:

Corollary 4.4.1 *Fix an adversarial prediction setting with outcome space*

$\mathcal{C} \subseteq [0,1]^d$ and a bucketing parameter m. There is an algorithm that against any sequence of outcomes $s^1, \ldots, s^T$ produces a transcript with predictions that are (α, m)-calibrated for:

$$\alpha = Bias(\pi^T, \mathcal{E}_{m-Cal}) \leq 5\sqrt{T(\ln(2d) + d\ln(m))}$$

Proof 38 *We just apply Theorem 24 for sufficiently small ϵ and $|\mathcal{E}| \leq m^d$.*

Finally, we can choose our choice of bucketing parameter m to optimize our bound from Theorem 25:

Corollary 4.4.2 *Fix any utility function $u : \mathcal{A} \times \mathcal{C}$ such that for each action $a \in \mathcal{A}$ $u(a, \cdot)$ is linear and L-Lipschitz in its second argument. Let $P : \mathcal{C} \to \mathcal{A}$ be any policy mapping "snapped" predictions $\tilde{s}$ to actions $\mathcal{A}$. If we choose $m = \left(\frac{T}{\alpha L}\right)^{\frac{1}{d+1}}$ Then for any transcript π^T whose predictions are (α, m)-calibrated, we have:*

$$\sum_{t=1}^{T} \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t} \left[u(BR(u, \tilde{s}^t), s^t)\right] \geq \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t} \left[u(P(\tilde{s}^t), s^t)\right] - 4T^{\frac{d}{d+1}} \cdot (\alpha L)^{\frac{1}{d+1}}$$

In particular, plugging in the bound for α from Corollary 4.4.1, we get:

$$\sum_{t=1}^{T} \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t} \left[u(BR(u, \tilde{s}^t), s^t)\right] \geq \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t} \left[u(P(\tilde{s}^t), s^t)\right] - \tilde{O}\left(T^{\frac{2d+1}{2d+2}}\right)$$

The statement we just proved most straightforwardly says something about the incentives that a downstream agent has (no matter what their utility function) to best respond to our predictions as if they were correct. If our predictions are calibrated, they will obtain higher utility just best responding to the predictions, compared to any other policy they could choose mapping predictions to actions. But what kinds of regret guarantees will these downstream agents have? It isn't hard to see that the downstream agents will not just have diminishing (external) regret, but will in fact have diminishing swap regret. The reason is because any benchmark sequence of actions that we could have obtained by counter-factually applying a swap function to their actions could have been directly obtained by some policy P — and we already know that no policy can consistently out perform the best response policy with respect to calibrated predictions.

Corollary 4.4.3 *Fix any utility function $u : \mathcal{A} \times \mathcal{C}$ such that for each action $a \in \mathcal{A}$, $u(a, \cdot)$ is linear and Lipschitz in its second argument. Fix a transcript π^T whose predictions are (α, m)-calibrated, with $m = \left(\frac{T}{\alpha L}\right)^{\frac{1}{d+1}}$ and α as in Corollary 4.4.1. Suppose an agent with utility function u takes action $a^t = BR(u, \tilde{s}^t)$ at each round t. Then the Agent has swap regret bounded by:*

$$\max_{\phi : \mathcal{A} \to \mathcal{A}} \sum_{t=1}^{T} \left(u(\phi(a^t), s^t) - u(a^t, s^t)\right) \leq \tilde{O}\left(T^{\frac{2d+1}{2d+2}}\right)$$

Proof 39 *We simply plug in the appropriate policy into Corollary 4.4.2. For any strategy modification rule, let* $P_\phi(\tilde{s}^t) = \phi(BR(u, \tilde{s}^t))$. *Then*

$$P_\phi(\tilde{s}^t) = \phi(BR(u, \tilde{s}^t)) = \phi(a^t)$$

and the bound follows directly from Corollary 4.4.2.

In some sense this is great — we've solved the problem we wanted to from the end of Section 4.2. We have a way of making predictions such that:

1. Any downstream agent, independently of their utility function, are actually incentivized to treat our predictions as correct, and
2. By doing so, they will obtain utility guarantees that are only stronger than swap regret guarantees.

But there are two serious problems:

1. The bounds we have proven in this section scale very badly with the dimension d: Even when $d = 1$, our bound is $\tilde{O}(T^{3/4})$ rather than $O(\sqrt{T})$, and it gets worse from there: for $d = 2$, the bound is $\tilde{O}(T^{5/6})$, and as the dimension increases the rate approaches the trivial bound of T exponentially fast with d. So these bounds are really only useful for very low dimensional problems.
2. Similarly, our generic multi-objective optimization algorithm when instantiated with the calibration problem requires solving a game whose strategy space for both players grows exponentially with d *and* whose number of objectives grows exponentially with d. Even if the regret bounds were useful in high dimensions, how would we efficiently make calibrated predictions?

4.5 Efficiently Making Unbiased Predictions

We now turn to the computational problem: How can we efficiently make predictions that obtain the bias guarantees that we proved (existentially) in Theorem 24? We'll start with 1-dimensional calibration — this is of course a special case (and the algorithm we give will not generalize to higher dimensions) — but has enough structure to give a very interesting algorithm that is worth seeing that avoids needing to solve an arbitrary minimax problem. We'll then turn to the general case in which we will solve a minimax problem — but one in which both the Learner and the Adversary have very large strategy spaces, and so solving it efficiently will require some cleverness.

4.5.1 One Dimensional (Multi)Calibration

Calibration will only be feasible for low dimensional problems — here we will give an algorithm for the case in which $d = 1$. But it is not be hard to extend 1-dimensional calibration to 1-dimensional "multicalibration", which we can think of as asking for calibration not just marginally, but on polynomially large collections of *subsequences* which can be defined by context and history. So that is what we'll do. First we recap some calibration definitions in the special case of $d = 1$.

Definition 43 (m-Bucketing) *The 1-dimensional m bucketing of the unit interval $[0, 1]$ is the partition:*

$$B(0) = \left[0, \frac{1}{m}\right) \quad B\left(\frac{1}{m}\right) = \left[\frac{1}{m}, \frac{2}{m}\right), \quad \ldots, \quad B\left(\frac{m-1}{m}\right) = \left[\frac{m-1}{m}, 1\right]$$

Observe that there are exactly m buckets in an m-bucketing, and the m-bucketing is a $1/m$-net for the unit interval.

We define multicalibration in terms of an m-bucketing:

Definition 44 *Let the prediction space $\mathcal{C} = [0, 1]$. Fix a collection of events $\mathcal{G}$ such that for each $G \in \mathcal{G}$, $G(t, x^t) \in [0, 1]$. We say that a transcript π^T corresponds to (α, m)-multicalibrated predictions with respect to $\mathcal{G}$ if it has $\mathcal{E}_{m,G}$-bias bounded by α for the set of events:*

$$\mathcal{E}_{m,\mathcal{G}} = \left\{E_{G,s'} = \mathbb{1}[s^t \in B(s')] \cdot G(t, x^t) : G \in \mathcal{G}, s' \in \left\{0, \frac{1}{m}, \ldots, \frac{m-1}{m}\right\}\right\}$$

Remark 4.5.1 *Just as subsequence regret asks for regret not just overall, but on many intersecting subsequences defined by subsequence selection functions, multicalibration asks for calibration not just overall but on many intersecting subsequences defined by intersecting "events", which are also represented by subsequence selection functions.*

We can encode multicalibration as a multiobjective optimization problem analogously to how we defined calibration. We will have an objective for each sign σ, prediction s', (and now) event G.

Definition 45 (r-Encoding (α, m)-Multicalibration as Multiobjective Optimization) *Given a 1 dimensional instance of the (α, m) Multicalibration problem with respect to events $\mathcal{G}$, we construct a $2|\mathcal{G}|m$-dimensional instance of the multi-objective optimization game as follows: At each round t:*

1. *The strategy space for the Learner is $\mathcal{A}^t = \Delta\{0, 1/r, 2/r, \ldots, 1\}$, the set of distributions over a finite $1/r$-net of $[0, 1]$.*
2. *The strategy space for the Adversary is $\mathcal{B}^t = [0, 1]$.*

3. For each $\sigma \in \{-1, 1\}$, $G \in \mathcal{G}$, and $s' \in \{0, 1/m, \ldots, (m-1)/m\}$ we construct a loss increment function:

$$\ell_{\sigma,G,s'}(p^t, s^t) = \sigma \cdot \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t}[\mathbb{1}[\hat{s}^t \in B(s')] \cdot G(t, x^t)(\hat{s}^t - s^t)]$$

Remark 4.5.2 *Observe that we allow the algorithm to play distributions over a $1/r$-net of the unit interval, whereas we measure calibration error with respect to a $1/m$ net. In general it will be important that we allow $r \gg m$. r here is a nuisance parameter — a discretization that we need to allow in order for the minimax theorem to apply, and we can take it to be as large as we want without affecting the final bounds. m on the other hand is a key parameter of the problem, effecting both the granularity of our final guarantees and the number of loss functions in our multiobjective optimization problem.*

We recall our generic multiobjective optimization algorithm, instantiated for the multicalibration problem:

Algorithm 17 Multicalibration via Multiobjective Optimization

for $t = 1$ to T **do**

Define the distribution $w^t \in \Delta[2|\mathcal{G}|m]$ as:

$$w^t_{\sigma,G,s'} \propto \exp\left(\frac{\eta}{2}\sum_{t'=1}^{t-1} \sigma \cdot \mathop{\mathbb{E}}_{\hat{s}^{t'} \sim p^{t'}}[\mathbb{1}[\hat{s}^{t'} \in B(s')] \cdot G(t', x^{t'})(\hat{s}^{t'} - s^{t'})]\right)$$

Define a zero-sum game in which the minimization player's action are $\mathcal{A}^t$, the maximization players actions are $\mathcal{B}^t$, and the utility function is:

$$u^t(p^t, s^t) = \sum_{\sigma,G,s'} w^t_{\sigma,G,s'} \sigma \cdot \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t}[\mathbb{1}[\hat{s}^t \in B(s')] \cdot G(t, x^t)(\hat{s}^t - s^t)]$$

Compute a minimax equilibrium strategy of this game p^t for the Learner and sample a forecast $\hat{s}^t$ from p^t.

We can read off bounds for our multicalibration error from our general theorem about unbiased prediction (Theorem 24)

Theorem 26 *Fix an r-encoding of an (α, m)-multicalibration problem with respect to a collection of events $\mathcal{G}$ (Definition 45). Running Algorithm 17 guarantees against any adversary that the (α, m)-multicalibration error with respect to $\mathcal{G}$ will be bounded by:*

$$\mathbb{E}[\alpha] \leq 4\sqrt{T \ln(2 \cdot |\mathcal{G}|m)} + \frac{T}{r}$$

Remark 4.5.3 *If we instantiate Theorem 27 with $r \geq \sqrt{T}$, then the multicalibration error we obtain is at most $5\sqrt{T \ln(2 \cdot |\mathcal{G}|m)}$.*

We now proceed to specialize the algorithm, taking advantage of the special structure of the 1-dimensional (multi)calibration problem to give a simpler algorithm that will not require that we solve a general minimax problem at every round. In fact, the algorithm we derive will be *almost* deterministic: it will make predictions that randomize between pairs of points that differ by at most $1/r$ from one another. To derive the algorithm, it will be helpful to first rewrite the utility function of the game that we need to solve to implement Algorithm 17. Because the utility function is linear in the learner's distribution p^t, it will be enough to write the utility as a function of the *pure strategies* $\hat{s}^t \in \{0, 1/r, \ldots, 1\}$.

Lemma 4.5.1 *Fix any $\hat{s}^t \in \{0, 1/r, \ldots, 1\}$. Let $\tilde{s}^t \in \{0, 1/m, 2/m, \ldots, (m-1)/m|\}$ be the unique point such that $\hat{s}^t \in B(\tilde{s}^t)$. Then we have that:*

$$u^t(\hat{s}^t, s^t) = \sum_{G \in \mathcal{G}} \underbrace{\left(w^{t+}_{G,\tilde{s}^t} - w^{t-}_{G,\tilde{s}^t}\right)}_{w^t_{G,\tilde{s}^t}} G(t, x^t)(\hat{s}^t - s^t)$$

where

$$w^{t^+}_{G,s'} \propto \exp\left(\frac{\eta}{2} \sum_{t'=1}^{t-1} \mathop{\mathbb{E}}_{\hat{s}^{t'} \sim p^{t'}} [\mathbb{1}[\hat{s}^{t'} \in B(s')] \cdot G(t', x^{t'})(\hat{s}^{t'} - s^{t'})]\right)$$

$$w^{t^-}_{G,s'} \propto \exp\left(\frac{\eta}{2} \sum_{t'=1}^{t-1} \mathop{\mathbb{E}}_{\hat{s}^{t'} \sim p^{t'}} [\mathbb{1}[\hat{s}^{t'} \in B(s')] \cdot G(t', x^{t'})(s^{t'} - \hat{s}^{t'})]\right)$$

Algorithm 18 A Sequential Multicalibration Algorithm

Given: m, r, and $\mathcal{G}$.
for $t = 1$ to T **do**
For each $s \in \left\{0, \frac{1}{m}, \ldots, \frac{m-1}{m}\right\}$, compute:

$$V_s = \sum_{G \in \mathcal{G}} w^t_{G,s} G(t, x^t)$$

if $V_s \geq 0$ for all s **then**
Predict $\hat{s}^t = 0$.
else if $V_s \leq 0$ for all s **then**
Predict $\hat{s}^t = 1$.
else
Choose $s \in \left\{\frac{1}{m}, \ldots, \frac{m-1}{m}\right\}$ such that

$$V_{s-1/m} \cdot V_s \leq 0$$

Choose $q \in [0, 1]$ such that

$$q \cdot V_s + (1 - q) \cdot V_{s-1/m} = 0$$

Predict $\hat{s}^t = s - \frac{1}{r}$ with probability $1 - q$ and **Predict** $\hat{s}^t = s$ with probability q.

Theorem 27 *Fix an r-encoding of an (α, m)-multicalibration problem with respect to a collection of events $\mathcal{G}$ (Definition 45). Algorithm 18 implements Algorithm 17 and so against any adversary obtains (α, m)-multicalibration error with respect to $\mathcal{G}$ bounded by:*

$$\mathbb{E}[\alpha] \leq 4\sqrt{T \ln(2 \cdot |\mathcal{G}|m)} + \frac{T}{r}$$

Proof 40 *We need to show that Algorithm 18 plays a strategy p^t that against any opponent action s^t guarantees utility at most $1/r$ in the game with utility function:*

$$u^t(\hat{s}^t, s^t) = \sum_{G \in \mathcal{G}} w^t_{G,\tilde{s}^t} G(t, x^t)(\hat{s}^t - s^t)$$

We consider the three cases in the algorithm.

1. ***Case 1****: $V_s \geq 0$ for all s. In this case, $\hat{s}^t = 0$ and we have:*

$$\begin{aligned} u(\hat{s}^t, s^t) &= u(0, s^t) \\ &= (0 - s^t) \cdot \sum_{G \in \mathcal{G}} w^t_{G,0} G(t, x^t) \\ &= -s^t \cdot V_0 \\ &\leq 0 \end{aligned}$$

where the last inequality follows from the fact that $-s^t \leq 0$ and $V_0 \geq 0$.

2. ***Case 2****: $V_s \leq 0$ for all s. In this case, $\hat{s}^t = 1$ and we have:*

$$\begin{aligned} u(\hat{s}^t, s^t) &= u(1, s^t) \\ &= (1 - s^t) \cdot \sum_{G \in \mathcal{G}} w^t_{G,\frac{m-1}{m}} G(t, x^t) \\ &= (1 - s^t) \cdot V_{\frac{m-1}{m}} \\ &\leq 0 \end{aligned}$$

where the last inequality follows from the fact that $1 - s^t \geq 0$ and $V_{\frac{m-1}{m}} \leq 0$.

3. ***Case 3****: $q \cdot V_s + (1-q) \cdot V_{s-1/m} = 0$. In this case, we can compute the expected utility:*

$$\begin{aligned} \mathbb{E}[u(\hat{s}^t, s^t)] &= q \cdot u(s, s^t) + (1 - q) u(s - 1/m, s^t) \\ &= q \cdot (s - s^t) V_s + (1 - q)(s - 1/r - s^t) V_{s - \frac{1}{m}} \\ &= (s - s^t) \left(q V_s + (1 - q) V_{s - \frac{1}{m}} \right) - \frac{(1 - q)}{r} V_{s - \frac{1}{m}} \\ &\leq 0 + \frac{1}{r} |V_{s - \frac{1}{m}}| \\ &\leq \frac{1}{r} \end{aligned}$$

which completes the proof in all cases.

Remark 4.5.4 *Just as calibration implies that (in the limit) it is a dominant strategy for any downstream agent with utility that is linear and Lipschitz in the state to choose actions using the best response policy, multicalibration with respect to a collection of events $\mathcal{G}$ guarantees that it will be a dominant strategy to use the best response policy not just overall, but on every subsequence defined by an event $G \in \mathcal{G}$.*

4.5.2 The General Case

We now consider the general problem of efficiently computing conditionally unbiased predictions in d dimensions against an adaptive adversary. Recall that given a collection of events $\mathcal{E}$, our goal is to bound:

$$\max_{E \in \mathcal{E}} \text{Bias}(\pi^T, E) = \max_{E \in \mathcal{E}} \left\| \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t} [E(t, \hat{s}^t, x^t) \cdot \left(\hat{s}^t - s^t\right)] \right\|_\infty$$

We recall our general algorithm for doing this:

Algorithm 19 Generic Algorithm for Conditionally Unbiased Prediction

for $t = 1$ to T **do**

Define the distribution $w^t \in \Delta[2|\mathcal{E}|d]$ as:

$$w^t_{\sigma,E,i} \propto \exp\left(\frac{\eta}{2} \sum_{t'=1}^{t-1} \sigma \cdot \mathop{\mathbb{E}}_{\hat{s}^{t'} \sim p^{t'}} [E(t', \hat{s}^{t'}, x^{t'})(\hat{s}^{t'}_i - s^{t'}_i)]\right)$$

Define a zero-sum game in which the minimization player's action are $\mathcal{A}^t = \Delta\mathcal{C}_\epsilon$, the maximization players actions are $\mathcal{C}$, and the utility function is:

$$u^t(p, s) = \sum_{\sigma,E,i} w^t_{\sigma,E,i} \sigma \cdot \mathop{\mathbb{E}}_{\hat{s} \sim p} [E(t, \hat{s}, x^t)(\hat{s}_i - s_i)]$$

Compute a minimax equilibrium strategy of this game p^t and sample $\hat{s}^t$ from p^t.

Elsewhere we have specialized algorithms of this sort, for subsequence regret, swap regret, and calibration. But in those cases, the "generic" algorithm was already polynomial time (it required solving a minimax problem with a linear program), and what we gained from specialization was getting a faster combinatorial algorithm, or some insight into the solution. Here though, it is not immediately apparent how to implement our generic algorithm in time polynomial in the dimension d. The difficulty is that the set of pure strategies for the learner, $\mathcal{C}_\epsilon$, has size $\Omega\left(\left(\frac{1}{\epsilon}\right)^d\right)$, exponentially large in d — and the strategy space $\mathcal{C}$ for the adversary is continuously large. Thus we cannot generically write down a polynomially sized linear program to solve this minimax problem, and if we hope to do so in time polynomial in d, we need to take advantage of some special structure that it has.

First let us recall one of the methods we derived for computing minimax equilibrium strategies: simulating play in which one player uses a "no-regret" algorithm, and the other player responds to the no-regret player's distribution at every round using a "value oracle". A value oracle takes as input an opponent strategy, and returns a strategy that is guaranteed to obtain at least the value of the game against the given opponent strategy:

Algorithm 20 Computing a Minimax Equilibrium: Value Oracle vs. No Regret

Given: A zero sum game $(\mathcal{A}_{\max}, \mathcal{A}_{\min}, u)$ satisfying the conditions of the minimax theorem, a Value oracle $Val : \mathcal{A}_{\min} \to \mathcal{A}_{\max}$ for Max, an online convex optimization algorithm OnlineConvex operating over action space $\mathcal{A}_{\min}$ and loss space $\{\ell = u(a, \cdot)\}_{a \in \mathcal{A}_{\max}}$ that promises regret $R(T)$ to every action $b \in \mathcal{A}_{\min}$ after T rounds, and an approximation parameter ϵ.
Let T be such that $R(T)/T \leq \epsilon$
for $t = 1$ to T **do**
  Get action b^t from OnlineConvex,
  Let $a^t = Val(b^t)$
  Feed loss $\ell^t = u(a^t, \cdot)$ to OnlineConvex.
Let $\bar{a} = \frac{1}{T}\sum_{t=1}^{T} a^t$
Return $\bar{a}$

We will see that we can efficiently implement this algorithm (in time polynomial in d and $|\mathcal{E}|$ by using *follow the perturbed leader* for the adversary, and implementing a Value oracle for the Learner by simply having them copy the adversary's strategy.

We start by rewriting an equivalent utility function for the Adversary that fits into the online linear optimization framework that is regret-equivalent to the original utility function u^t.

Lemma 4.5.2 *Define the cost function:*

$$c^t(p, s) = \sum_{\sigma, E, i} w^t_{\sigma,E,i} \mathop{\mathbb{E}}_{\tilde{s} \sim p}[E(t, \tilde{s}, x^t)s_i] = \langle s, c^t(p)\rangle$$

where $c^t(p) \in \mathbb{R}^d$ is defined as:

$$c^t_i(p) = \sum_{\sigma, E} w^t_{\sigma,E,i} \mathop{\mathbb{E}}_{\tilde{s} \sim p}[E(t, \tilde{s}, x^t)]$$

Then if on transcript π^T the adversary has regret α with respect to minimizing costs c^t:

$$\sum_{t=1}^{T} \langle s^t, c^t(p^t)\rangle \leq \min_{s \in \mathcal{C}} \sum_{t=1}^{T} \langle s, c^t(p^t)\rangle + \alpha$$

they also have regret α with respect to maximizing utilities u^t:

$$\sum_{t=1}^{T} u^t(p^t, s^t) \geq \max_{s \in \mathcal{C}} \sum_{t=1}^{T} u^t(p^t, s) - \alpha$$

Proof 41 *First we observe that we can write the utility function:*

$$u^t(p, s) = \underbrace{\sum_{\sigma, E, i} w^t_{\sigma,E,i} \mathop{\mathbb{E}}_{\hat{s} \sim p}[E(t, \hat{s}, x^t)\hat{s}_i]}_{\hat{u}^t(p)} - c^t(p, s)$$

Importantly, the first term is independent of the Adversary's action s. Therefore we have:

$$\begin{aligned}
\max_{s\in\mathcal{C}} \sum_{t=1}^{T} u^t(p^t, s) &= \max_{s\in\mathcal{C}} \sum_{t=1}^{T} \left(\hat{u}^t(p^t) - c^t(p^t, s)\right) \\
&= \sum_{t=1}^{T} \hat{u}^t(p^t) - \min_{s\in\mathcal{C}} \sum_{t=1}^{T} c^t(p^t, s) \\
&\leq \sum_{t=1}^{T} \hat{u}^t(p^t) - \sum_{t=1}^{T} c^t(p^t, s^t) + \alpha \\
&= \sum_{t=1}^{T} u^t(p^t, s^t)
\end{aligned}$$

Thus:

$$\sum_{t=1}^{T} u^t(p^t, s^t) \geq \max_{s\in\mathcal{C}} \sum_{t=1}^{T} u^t(p^t, s) - \alpha$$

as claimed.

Since obtaining no-regret to linear costs of the form $\langle s, c^t(p)\rangle$ is just an online linear optimization problem, we can solve it using follow-the-perturbed-leader. We can read off the regret bounds that the Adversary can obtain in this setting from our bound on the regret for Follow the Perturbed Leader:

Lemma 4.5.3 *In the minimax equilibrium computation, the Adversary using Follow the Perturbed Leader can obtain regret to the best action in $\mathcal{C}$:*

$$R(T) \leq \sqrt{8CRT}$$

where $C = \max_{s\in\mathcal{C}} ||s||_1$ and $R = \max_{s\in\mathcal{C}} ||s||_\infty$.

Proof 42 *We apply the bound from Theorem 3, observing that $||c^t(p)||_1 \leq 1$ since the weights $w_{\sigma,E,i}$ form a probability distribution, and hence*

$$\max_{s\in\mathcal{C}, c^t(p)\in[0,1]^d} \langle s, c^t(p)\rangle \leq \max_{s\in\mathcal{C}} ||s||_\infty$$

What about implementing a value oracle? The utility function is *not* linear in the (pure) strategies of the Learner because the events E can be arbitrary, and so it is not clear how we would efficiently implement a "best response" oracle. However, in this case it turns out that implementing a value oracle is much easier: We simply copy (the expectation of) the mixed strategy deployed by the Adversary.

Lemma 4.5.4 *For the game with utility function*

$$u^t(p, s) = \sum_{\sigma,E,i} w^t_{\sigma,E,i}\sigma \cdot \mathop{\mathbb{E}}_{\hat{s}\sim p} [E(t, \hat{s}, x^t)(\hat{s}_i - s_i)]$$

given any distribution $q \in \Delta\mathcal{C}$ over strategies of the adversary, we can implement a value oracle as:

$$Val(q) = \mathop{\mathbb{E}}_{s\sim q}[s]$$

Proof 43 *We recall that the value of the bias game is* 0*. Thus we need to show that for any $q \in \Delta\mathcal{C}$, if $\hat{s} = \mathbb{E}_{s\sim q}[s]$, then $\mathbb{E}_{s\sim q}[u(\hat{s}, s)] \leq 0$. We calculate:*

$$\begin{aligned}
\mathop{\mathbb{E}}_{s\sim q}[u(\hat{s}, s)] &= \mathop{\mathbb{E}}_{s\sim q}\left[\sum_{\sigma,E,i} w^t_{\sigma,E,i}\sigma \cdot E(t, \hat{s}, x^t)(\hat{s}_i - s_i)\right] \\
&= \sum_{\sigma,E,i} w^t_{\sigma,E,i}\sigma \cdot E(t, \hat{s}, x^t)(\hat{s}_i - \mathop{\mathbb{E}}_{s\sim q}[s_i]) \\
&= 0
\end{aligned}$$

which completes the proof.

Thus we can instantiate the Value-Oracle vs. No-Regret Dynamics as follows.

Algorithm 21 Efficiently Computing a Minimax Equilibrium of the Unbiased Predictions Game

Given: An outcome space $\mathcal{C}$, a collection of events $\mathcal{E}$, and an approximation parameter ϵ.
Let $C = \max_{s\in\mathcal{C}} ||s||_1$ and $R = \max_{s\in\mathcal{C}} ||s||_\infty$. **Let** $T = \frac{8CR}{\epsilon^2}$.
Initialize a copy of Follow the Perturbed Leader (FTPL) to solve a d-dimensional online linear optimization problem over $\mathcal{C}$.
for $t = 1$ to T **do**
  Obtain distribution q^t from FTPL.
  Let $\hat{s}^t = \mathbb{E}_{s\sim q}[s]$.
  Feed loss $c^t \in \mathbb{R}^d$ to FTPL where:

$$c_i = \sum_{\sigma,E} w^t_{\sigma,E,i} E(t, \hat{s}^t, x^t)$$

Return the distribution $p^t \in \Delta\mathcal{C}$ that is uniform over $\{\hat{s}^1, \ldots, \hat{s}^T\}$

We then have as a corollary of Theorem 9 that we can compute an ϵ-approximate equilibrium for the game we need to in order to make conditionally unbiased predictions with respect to $\mathcal{C}$, in time polynomial in $|\mathcal{E}|, d$, and $1/\epsilon$.

Corollary 4.5.1 *Assuming we can solve linear optimization problems over $\mathcal{C}$, Algorithm 21 outputs an ϵ-approximate equilibrium of the conditional bias game in time polynomial in $d, |\mathcal{E}|$, and $1/\epsilon$.*

Remark 4.5.5 *Note that to implement the value oracle we have assumed that we can compute $\hat{s}^t = \mathbb{E}_{s\sim q}[s]$, whereas what FTPL gives us is an efficient (given a linear optimization oracle over $\mathcal{C}$) implementation of a sampling oracle for distribution q. For some common geometries for $\mathcal{C}$ (e.g. the rectangle or the simplex) it is not hard to find an expression for the expectation in closed form. But otherwise, we need to approximate this quantity by sampling. We only need to approximate this quantity up to error $O(\epsilon)$ in each coordinate, which we can do with high probability with polynomially many in $1/\epsilon$ many samples using standard concentration arguments.*

The upshot is that we can *efficiently* make predictions in d dimensions that are unbiased with respect to polynomially many events $\mathcal{E}$. This is not enough to give us efficient algorithms for full calibration (which requires unbiasedness with respect to exponentially many in d events) — but as we will see, is enough for many of our goals.

4.6 Predicting for No-Swap-Regret Play

Ok! Now that we know how to efficiently make predictions that are unbiased subject to a polynomial number of conditioning events, lets put this technology to work on the problem we initially set out to solve: making predictions so that many downstream agents will obtain strong guarantees when they best respond to them. Recall that in Section 4.2 we showed how to guarantee that arbitrary downstream agents obtain no *external* regret — we simply added noise to the empirical history of play, which "tricked" best responding agents into playing follow the perturbed leader. In Section 4.4, we showed that calibrated predictions are sufficient to guarantee arbitrary downstream agents no *swap* regret if they best respond to them — but unfortunately at rates that degraded exponentially with the dimension d of the problem, and that did not even obtain the correct $O(\sqrt{T})$ rates when the dimension $d = 1$. We also do not have a computationally efficient (in d) algorithm for producing calibrated forecasts in d dimensions. In this section, we will show how to produce d-dimensional predictions $\hat{s}^t \in \mathcal{C} \subseteq \mathbb{R}^d$ that guarantee that any downstream agent with a utility function $u : \mathcal{A} \times \mathcal{C} \to [0, 1]$, with $u \in \mathcal{U}$ who bests responds to our predictions swap regret tending to 0 at a rate scaling like $O(\sqrt{T \log(d|\mathcal{U}|)})$. Moreover, we will be able to make our predictions in time polynomial in d, $|\mathcal{A}|$, and $|\mathcal{U}|$. This efficiently gives (up to log terms) nearly optimal swap regret to a collection of downstream decision makers — although compared to results from previous sections, our guarantees no longer hold for *all* downstream decision makers, but only for those with utility functions from a fixed-up-front set $\mathcal{U}$, whose cardinality we depend linearly on in our running time and logarithmically on in our regret bounds. The algorithm will be the one we have already developed: we will simply make predictions that are

conditionally unbiased conditional on events that are defined in terms of the best response correspondence of the utility functions in $\mathcal{U}$.

Definition 46 (Best Response Events) *Fix a state space $\mathcal{C} \subseteq \mathbb{R}^d$ and an action set $\mathcal{A}$. Given a utility function $u : \mathcal{A} \times \mathcal{C} \to [0,1]$, for each $a \in \mathcal{A}$, define the best-response event:*

$$E_{u,a}(s) = \mathbb{1}[a = BR(u,s)]$$

where we recall that:

$$BR(u,s) = \arg\max_{a \in \mathcal{A}} u(a,s)$$

Here we assume that ties are broken consistently (e.g. by choosing the lexicographically first action) when the arg max *is not unique. Given a collection $\mathcal{U}$ of such utility functions, we define the collection of events:*

$$\mathcal{E}_{\mathcal{U}} = \{E_{u,a}\}_{u \in \mathcal{U}, a \in \mathcal{A}}$$

First we note that this isn't too many events: Given an action space $\mathcal{A}$ and a set of utility functions $\mathcal{U}$, we have that $|\mathcal{E}_{\mathcal{U}}| = |\mathcal{U}| \cdot |\mathcal{A}|$ — so it is polynomial in the number of utility functions and actions, and independent of the dimension d of the state space. Next we observe that if a sequence of predictions is unbiased with respect to $\mathcal{E}_{\mathcal{U}}$, then it guarantees no swap regret for every $u \in \mathcal{U}$ (assuming they follow the best response policy).

The intuition is very simple. Consider the subsequence of rounds on which such an agent plays a particular action (say action 1). Would they have preferred to play another action (say action 2) instead, in hindsight? Well, this subsequence of rounds corresponds to the event $E_{u,1}$, which is one of our conditioning events — and so the sequence of our predictions is correct on average over this sequence. The utility function is linear, which means that the utility that the agent gets for playing action 1 is what he thought it would be, given the predictions. It also means that the utility that the agent *thought he would have gotten* from playing action 2 is what he thought it would be. And the reason he played action 1 over action 2 is because he thought action 1 would have higher payoff — as it must have had, on average, over this sequence. Thus he has no swap regret. We now formalize this argument.

Theorem 28 *Fix a set of utility functions $\mathcal{U}$ over state space $\mathcal{C}$ and action space $\mathcal{A}$, such that each utility function $u \in \mathcal{U}$ is such that $u : \mathcal{A} \times \mathcal{C} \to [0,1]$ is linear and L-Lipschitz in its second argument. Fix a transcript π^T of predictions $\hat{s}^1, \ldots \hat{s}^T$ that has bias at most α conditional on the events $\mathcal{E}_{\mathcal{U}}$. Suppose at each round, an agent with utility function $u \in \mathcal{U}$ takes an action $a^t = BR(u, \hat{s}^t)$. Then this agent has swap regret bounded by:*

$$\max_{\phi : A \to A} \sum_{t=1}^{T} \left(\mathop{\mathbb{E}}_{\hat{s}^t \sim p^t} [u(\phi(a^t), s^t) - u(a^t, s^t)] \right) \leq 2|\mathcal{A}|\alpha L$$

Towards proving this, it will be useful to first observe that if an action is a best response to states s^1 and s^2, then it is also a best response to any linear combination of these states.

Lemma 4.6.1 *Fix any two $s^1, s^2 \in \mathbb{R}^d$ and any two $c_1, c_2 \in \mathbb{R}^{\geq 0}$. Suppose that $a = BR(u, s^1)$ and $a = BR(u, s^2)$. Then: $a = BR(u, c_1 s^1 + c_2 s^2)$.*

Proof 44 *Fix any other action a'. We have that:*

$$\begin{aligned} u(a, c_1 s^1 + c_2 s^2) - u(a', c_1 s^1 + c_2 s^2) &= c_1(u(a, s^1) - u(a', s^1)) + c_2(u(a, s^2) - u(a', s^2)) \\ &\geq 0 \end{aligned}$$

as a is a best response to both s^1 and s^2. Moreover, if a is the lexicographically first best response to both s^1 and s^2, it is also the lexicographically first best response to $c_1 s^1 + c_2 s^2$.

Proof 45 (Proof of Theorem 28) *We now formalize our intuition. Fix any utility function $u \in \mathcal{U}$ and swap function $\phi : A \to A$.*

$$\begin{aligned} \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t}[u(a^t, s^t)] &= \sum_{t=1}^{T} \sum_{a \in \mathcal{A}} \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t}[\mathbb{1}[a = BR(u, \hat{s}^t)] u(a, s^t)] \\ &= \sum_{a \in \mathcal{A}} u\left(a, \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t}[E_{u,a}(\hat{s}^t)] s^t\right) \\ &\geq \sum_{a \in \mathcal{A}} \left(u\left(a, \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t}[E_{u,a}(\hat{s}^t)\hat{s}^t]\right) - \alpha L\right) \\ &\geq \sum_{a \in \mathcal{A}} u\left(\phi(a), \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t}[E_{u,a}(\hat{s}^t)\hat{s}^t]\right) - \alpha|\mathcal{A}|L \\ &\geq \sum_{a \in \mathcal{A}} \left(u\left(\phi(a), \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t}[E_{u,a}(\hat{s}^t)] s^t\right) - \alpha L\right) - \alpha|\mathcal{A}|L \\ &= \sum_{t=1}^{T} \sum_{a \in \mathcal{A}} \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t}[\mathbb{1}[a = BR(u, \hat{s}^t)] u(\phi(a), s^t)] - 2\alpha|\mathcal{A}|L \\ &= \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\hat{s}^t \sim p^t}[u(\phi(a^t), s^t)] - 2\alpha|\mathcal{A}|L \end{aligned}$$

Here the 2nd and and 3rd equalities follow from the linearity of the objective function, the first and third inequalities follow from the α bias condition together with the L-Lipschitzness of the objective function, and the 2nd inequality follows from the definition of the best response function and Lemma 4.6.1

Plugging in the bound on α that we get from running Algorithm 21 on the collection of events $\mathcal{E}_{\mathcal{U}}$, we obtain the following corollary:

Corollary 4.6.1 *Fix a set of utility functions $\mathcal{U}$ over state space $\mathcal{C}$ and action space $\mathcal{A}$, such that each utility function $u \in \mathcal{U}$ is linear and L-Lipschitz in its second argument.*

There is an algorithm running in time polynomial in d, $|\mathcal{U}|$, and $|\mathcal{A}|$ that produces forecasts $\hat{s}^t$ such that for any agent with utility function $u \in \mathcal{U}$ who takes an action $a^t = BR(u, \hat{s}^t)$, then this agent has swap regret bounded by:

$$\max_{\phi:A\to A} \sum_{t=1}^{T} \left(\mathop{\mathbb{E}}_{\hat{s}^t \sim p^t} [u(\phi(a^t), s^t) - u(a^t, s^t)] \right) \leq 2|\mathcal{A}|L\sqrt{T \ln(2d|\mathcal{U}||\mathcal{A}|)}$$

4.7 Obtaining No-Subsequence-Regret in Online Combinatorial Optimization

In Section 4.6 we showed how to efficiently make predictions for downstream agents with utility functions in $\mathcal{U}$ that cause each of them to obtain (up to a $\log |\mathcal{U}|$ term) swap regret bounds that are as good as they could have obtained had they run their own swap regret algorithm — but now, all that we require of them is that they best respond to our predictions. We can think of this as a way of coordinating agents towards good outcomes even when the individual agents don't have the sophistication to run complicated algorithms. But if all we want to do is provide a swap regret guarantee for a single agent capable of running an algorithm we give them, we haven't in fact improved over the basic swap regret algorithm we derived in Section 3.3.2.1.

In this section we show how to use the machinery we have developed for making conditionally unbiased predictions to give *new* efficient learning algorithms: in particular, how to get *subsequence regret* (as defined in Section 3.3) in d-dimensional combinatorial optimization settings in time polynomial in d. In a d-dimensional combinatorial optimization setting, there are d base actions, but the learner can play *compound* actions that are subsets of these d base actions. Each day a gain or a cost is realized for each of the base actions, and the gain or cost of a compound action is the gain or cost of the base actions included in the compound action. The online shortest paths problem is a special case of this, and more generally these problems are instances of online linear optimization. Importantly, the number of *actions* in an online combinatorial optimization setting can be as large as 2^d. We say in Section 1.4 how to efficiently (in d) obtain no *external* regret in online combinatorial optimization settings (and more generally online linear optimization settings) — but our algorithms for subsequence regret from Section 3.3 depended polynomially on the number *of actions* in the game, and hence would scale with 2^d. We will show that to obtain no subsequence regret with respect to a collection of events $\mathcal{E}$, it will suffice to best-respond to forecasts of the base-action gains that are unbiased subject to a collection of conditioning events of size

$d \cdot |\mathcal{E}|$, and so can be done in time polynomial in d and $|\mathcal{E}|$. Of course, if we have many downstream decision makers who have utility functions from $\mathcal{U}$, we can include $d \cdot |\mathcal{E}|$ conditioning events defined in terms of the utility functions corresponding to each $u \in \mathcal{U}$ and get downstream subsequence regret guarantees for all of the agents — but for clarity, we will focus on a single utility function, and view our task as an algorithm design problem for a single agent.

Definition 47 (Online Combinatorial Optimization) *An online combinatorial optimization problem is defined by a set of d base actions $\mathcal{B}$ and a set of actions $\mathcal{A} \in 2^{\mathcal{B}}$ such that each action $a \in \mathcal{A}$ is a subset of $\mathcal{B}$: $a \subseteq \mathcal{B}$. In rounds $t = 1, \ldots, T$:*

1. *The learner chooses a distribution over actions $p^t \in \Delta\mathcal{A}$,*
2. *The adversary chooses a vector of gains $g^t \in [0,1]^d$, indexed by the base actions $\mathcal{B}$.*
3. *The learner obtains utility*

$$u(p^t, g^t) = \mathop{\mathbb{E}}_{a^t \sim p^t}[u(a^t, g^t)] = \mathop{\mathbb{E}}_{a^t \sim p^t}\left[\sum_{b \in a^t} g^t_a\right]$$

We now recall the definition of subsequence regret from Section 3.3, which is defined by a collection of subsequence selection functions $\mathcal{E}$ (which are very similar objects to the "events" that we speak of in conditionally unbiased predictions, but are functions of an Agent's actions in $\mathcal{A}$ rather than *predictions* in $\mathcal{C}$.) A subsequence selection function $E : [T] \times \mathcal{A} \times \mathcal{X} \to \{0,1\}$ maps time (and history), the action selected by the Agent/Learner, and context to an indicator in $\{0,1\}$ expressing if the current round is a member of the subsequence or not. An agent with utility function $u(p^t, g^t)$ who chooses (distributions over) actions $p^1, \ldots, p^T$ when the vectors of gains chosen by the adversary are realized as $g^1, \ldots, g^T$ will have subsequence regret to action $a \in \mathcal{A}$ defined as:

$$Reg(\pi^T, E, a) = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{a^t \sim p^t}\left[E(t, a^t, x^t) \cdot (u(a, g^t) - u(a^t, g^t))\right]$$

We say that the learner has expected $\mathcal{E}$ subsequence regret bounded by α if:

$$\max_{E \in \mathcal{E}, a \in \mathcal{A}} Reg(\pi^T, E, a) \leq \alpha$$

Note that in an online combinatorial optimization setting, subsequence regret is asking for regret to all of the possibly $\Omega(2^d)$ actions in $\mathcal{A}$, over each of the subsequences $E \in \mathcal{E}$. Our goal will be to efficiently achieve this for any polynomial (in d) collection of subsequence indicator functions $\mathcal{E}$.

Conditioning on the events $E_{u,a}$ that we used in Section 4.6 is no longer feasible, because there are too many such events — one for each of the possibly

$\Omega(2^d)$ actions in $\mathcal{A}$. Instead we need to take advantage of the linear structure of online combinatorial optimization. Towards that end, we define events in terms of the *base actions* $b \in \mathcal{B}$ and their inclusion in the best response.

Definition 48 *Given a vector of gains g define*

$$BR(g) = \arg\max_{a \in \mathcal{A}} u(a, g) = \arg\max_{a \in \mathcal{A}} \sum_{b \in a} g_a^t$$

breaking ties lexicographically. For each $b \in \mathcal{B}$ define the event:

$$E_b(g) = \mathbb{1}[b \in BR(g)]$$

Definition 49 *Given a collection of subsequence selection functions $\tilde{\mathcal{E}}$, define the collection of events:*

$$\mathcal{E}(\tilde{\mathcal{E}}, \mathcal{B}) = \{E_{\tilde{E},b}(t, g, x^t) = \tilde{E}(t, BR(g), x^t) \cdot E_b(g)\}_{\tilde{E} \in \tilde{\mathcal{E}}, b \in \mathcal{B}}$$

This collection of events scales only with $d \cdot |\tilde{\mathcal{E}}|$, but will suffice to give us no subsequence regret over the subsequences in $\tilde{\mathcal{E}}$ and all of the (exponentially many in d) actions in $\mathcal{A}$.

Theorem 29 *Fix a d-dimensional online combinatorial optimization problem and a collection of subsequence indicator functions $\tilde{\mathcal{E}}$. Let the state space $\mathcal{C} = [0, 1]^d$ be the set of feasible gains. Fix a transcript π^T of predictions $\hat{g}^1, \ldots, \hat{g}^T$ that has bias at most α conditional on the events in $\mathcal{E}(\tilde{\mathcal{E}}, \mathcal{B}) \cup \tilde{\mathcal{E}}$. Suppose at each round the Learner chooses the action $a^t = BR(\hat{g}^t)$. Then the agent has $\tilde{E}$-subsequence regret bounded by:*

$$\max_{\tilde{E} \in \tilde{\mathcal{E}}, a \in \mathcal{A}} Reg(\pi^T, \tilde{E}, a) \leq 2\alpha d$$

Proof 46 *Fix a subsequence $\tilde{E} \in \tilde{\mathcal{E}}$ and an action $a \in \mathcal{A}$. We can compute:*

$$\begin{aligned}
\sum_{t=1}^{T} \tilde{E}(t, a^t, x^t) \cdot u(a^t, g^t) &= \sum_{t=1}^{T} \left(\tilde{E}(t, a^t, x^t) \cdot \sum_{b \in a^t} g_b^t \right) \\
&= \sum_{b \in \mathcal{B}} \sum_{t=1}^{T} \tilde{E}(t, a^t, x^t) \cdot E_b(\hat{g}^t) \cdot g_b^t \\
&\geq \sum_{b \in \mathcal{B}} \left(\sum_{t=1}^{T} \tilde{E}(t, a^t, x^t) \cdot E_b(\hat{g}^t) \cdot \hat{g}_b^t - \alpha \right) \\
&= \sum_{b \in \mathcal{B}} \sum_{t=1}^{T} \tilde{E}(t, a^t, x^t) \cdot E_b(\hat{g}^t) \cdot \hat{g}_b^t - \alpha \cdot d \\
&= \sum_{t=1}^{T} \tilde{E}(t, a^t, x^t) \cdot u(BR(\hat{g}^t), \hat{g}^t) - \alpha \cdot d \\
&\geq \sum_{t=1}^{T} \tilde{E}(t, a^t, x^t) \cdot u(a, \hat{g}^t) - \alpha \cdot d \\
&= \sum_{b \in a} \sum_{t=1}^{T} \tilde{E}(t, a^t, x^t) \cdot \hat{g}_b^t - \alpha d \\
&\geq \sum_{b \in a} \sum_{t=1}^{T} \tilde{E}(t, a^t, x^t) \cdot g_b^t - 2\alpha d \\
&= \sum_{t=1}^{T} \tilde{E}(t, a^t, x^t) \cdot u(a, g^t) - 2\alpha \cdot d
\end{aligned}$$

which is what we wanted.

Once again, we can plug in the bound on α that we get from running algorithm 21 on the collection of events $\mathcal{E}(\tilde{\mathcal{E}}, \mathcal{B}) \cup \tilde{\mathcal{E}}$ to obtain the following corollary:

Corollary 4.7.1 *Fix a d-dimensional online combinatorial optimization problem and a collection of subsequence indicator functions $\tilde{\mathcal{E}}$. Let the state space $\mathcal{C} = [0,1]^d$ be the set of feasible gains. There is an algorithm that in time polynomial in $|\tilde{\mathcal{E}}|$ and d produces a transcript of actions a^t such that against any adversarially realized sequence of gains has expected $\tilde{E}$-subsequence regret bounded by:*

$$\max_{\tilde{E} \in \tilde{\mathcal{E}}, a \in \mathcal{A}} Reg(\pi^T, \tilde{E}, a) \leq 2d \cdot \sqrt{T \ln(2|\tilde{\mathcal{E}}|(d+1))}$$

Note that because the action space $\mathcal{A}$ is exponentially large in d, this is *not* something that we knew how to do before.

4.8 Predicting Label Probabilities with "Transparent Coverage"

Before we close out this Chapter we'll give one more application of our unbiased prediction technology.

Suppose we are trying to solve a k-class classification problem: At each round t, we observe features x^t and need to predict a label $y^t \in \{1, \ldots, k\}$ that could take k discrete values. For example, this could be an image classification task: x^t would in this case be the pixel representation of an image, and the label set would consist of the names of k possible objects that might be in the image.

Classification problems can be difficult, and so any classification technology must be prepared to make mistakes. How should we express the uncertainty of a classification technology? One was is by producing a *prediction set* rather than a point prediction: a *set* of labels $S^t \subseteq [k]$ that is intended to contain ("cover") the true label y^t with some target probability, say 90%.

How should we produce a prediction set? Many kinds of classifiers, given features x, actually produce *scores* $s_i^t(x) \in [0, 1]$ for each label $i \in [k]$. Often these scores "look like" probabilities in the sense that $\sum_{i=1}^k s_i^t(x) = 1$, and higher scores are supposed to connote more likely labels: the standard way to turn such scores into point predictions is to predict the label that has the highest score.

If the labels $s_i^t(x)$ somehow represented "true label probabilities" $\Pr[y^t = i|x^t]$[1], then there would be a simple method to produce the smallest prediction set that covered the label 90% of the time: Simply produce the set S^t that consists of the shortest prefix of labels, in descending order by their probabilities $s_i^t(x)$, such that their cumulative probability sums to at least 90%.

More generally, if the scores $s_i^t(x)$ represented true probabilities, then any algorithm mapping vectors of probabilities to prediction sets $S : [0, 1]^k \to 2^{[k]}$ would, for any sequence of examples, cover the true label at a rate that could be "read off" from the probabilities. We call this a "transparent coverage" guarantee. Of course, the difficulty will be that it is not possible to produce scores that represent "true" probabilities.

Definition 50 *Fix any method for sequentially producing class scores $s^t : \mathcal{X} \times \Pi^{<t} \to [0, 1]^k$ and any method S for mapping class scores to prediction sets $S : [0, 1]^k \to 2^{[k]}$. Fix any transcript π^T including a sequence of examples (x^t, y^t). Then the apparent coverage of S is:*

$$\tilde{\text{Cov}}(\pi^T) = \frac{1}{T}\sum_{t=1}^{T} \mathop{\mathbb{E}}_{s^t, S}\left[\sum_{i \in S(s^t(x^t, \pi^{<t}))} s_i^t(x^t, \pi^{<t})\right]$$

[1] Don't think too hard about whether such probabilities actually make sense or you might become a philosopher

The actual coverage of S is:

$$\mathrm{Cov}(\pi^T) = \frac{1}{T}\sum_{t=1}^{T} \mathop{\mathbb{E}}_{s^t,S}\left[\mathbb{1}[y^t \in S(s^t(x^t, \pi^{<t}))]\right]$$

We say that S has α-transparent coverage with respect to scores s^t if for every transcript π^T:

$$\left|\tilde{\mathrm{Cov}}(\pi^T) - \mathrm{Cov}(\pi^T)\right| \leq \alpha$$

If S satisfies a transparent coverage guarantee with respect to scores s^t, that means that the coverage you would expect of the prediction sets $S(s^t(x))$, were the scores $s^t(x)$ real probabilities, is in fact the coverage you get — at least as averaged over all of the T examples. So, for example, if S is defined so that it always gets (say) 90% coverage when the scores $s^t_i(x)$ represent true class probabilities $\Pr[y^t = i|x^t]$, then S will still get 90% coverage.

What we will show is that if our class scores $s^t_i(x)$ are unbiased subject to the *label selection events* $\mathcal{E}_S$ defined with respect to a prediction set algorithm $S : [0,1]^k \to 2^{[k]}$ then S will satisfy a transparent coverage guarantee with respect to scores s^t.

Definition 51 *Fix a prediction set algorithm $S : [0,1]^k \to 2^{[k]}$. For each label $i \in [k]$ define the event:*

$$E_{S,i}(s^t) = \mathbb{1}[i \in S(s^t)]$$

And let $\mathcal{E}_S = \{E_{S,i}\}_{i\in[k]}$.

It will be helpful for us, when discussing the bias of our k dimensional predictions s^t, to encode labels y^t as k-dimensional vectors $e^t(y^t)$ defined such that $e^t_i(y^t) = 1$ for $i = y^t$ and $e^t_j = 0$ for all $j \neq y^t$.

Theorem 30 *Fix a prediction set algorithm $S : [0,1]^k \to 2^{[k]}$. Fix a transcript π^T on which the predicted scores $s^1, \ldots, s^T$ have at most α bias conditional on the events $\mathcal{E}_S$ (where their prediction target is the labels $y^1, \ldots, y^T$ represented as k-dimensional indicator variables $e^t(y_t)$). Then S has $\alpha \cdot k/T$-transparent coverage:*

$$\left|\tilde{\mathrm{Cov}}(\pi^T) - \mathrm{Cov}(\pi^T)\right| \leq \frac{\alpha \cdot k}{T}$$

Proof 47 *We can compute:*

$$\begin{aligned}
\tilde{\text{Cov}}(\pi^T) &= \frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{s^t,S}\left[\sum_{i\in S(s^t)} s_i^t\right] \\
&= \frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{s^t,S}\left[\sum_{i\in[k]} E_{S,i}(s^t)s_i^t\right] \\
&= \frac{1}{T}\sum_{i\in[k]}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{s^t,S}\left[E_{S,i}(s^t)s_i^t\right] \\
&\leq \frac{1}{T}\sum_{i\in[k]}\left(\sum_{t=1}^{T}\mathop{\mathbb{E}}_{s^t,S}\left[E_{S,i}(s^t)e_i^t(y^t)+\alpha\right)\right] \\
&= \frac{1}{T}\left(\sum_{t=1}^{T}\mathop{\mathbb{E}}_{s^t,S}[\mathbb{1}[y^t\in S(s^t)]]+\alpha k\right) \\
&= \text{Cov}(\pi^T)+\frac{\alpha k}{T}
\end{aligned}$$

We also have symmetrically that $\tilde{\text{Cov}}(\pi^T) \geq \text{Cov}(\pi^T) - \frac{\alpha k}{T}$*, which proves the claim.*

As usual, we can plug in the bound on α that we get from running Algorithm 21 to make predictions over the simplex $\Delta[k]$ that are unbiased with respect to $\cup_{S\in\mathcal{S}}\mathcal{E}_S$ for any collection of prediction set algorithms $S : [0,1]^k \to 2^{[k]}$ to obtain the following corollary:

Corollary 4.8.1 *Fix any collection $\mathcal{S}$ of prediction set algorithms $\mathcal{S}$. There is an algorithm running in time polynomial in k and $|\mathcal{S}|$ per time-step that makes predictions s^t such that simultaneously for each $S \in \mathcal{S}$, S has α-transparent coverage with respect to s^t for:*

$$\alpha \leq k \cdot \frac{\sqrt{\ln(2|\mathcal{S}|k)}}{T}$$

This gives us a way to obtain class scores that satisfy transparent coverage guarantees for any collection of prediction set algorithms. But what about accuracy? The algorithms we have given ignore the context x^t, so cannot in general be very good predictive algorithms. It turns out this is an easy fix. Suppose we have some trained model $f : \mathcal{X} \to [0,1]^k$ that produces class scores as a function of context. If we additionally ask that our forecasts are *multicalibrated* (marginally, in each dimension i) with respect to the level sets of f (which we can do efficiently, by asking that our forecasts have low bias with respect to the cartesian product of level sets of f and level sets of our predictor), then our predictions will perform as well as f as measured by squared error, cross-entropy-loss, or indeed any other "proper scoring rule".

Bibliographic Notes and Further Reading

The idea of making predictions so that downstream decision-makers who best respond to those predictions have low external regret is explored by Kleinberg et al. [2023], which they call "U-Calibration". The follow-the-perturbed-algorithm like algorithm we give for this problem is taken from Kleinberg et al. [2023]. The algorithm we give for efficiently making d-dimensional predictions that are unbiased with respect to a polynomial collection of conditioning events, together with the applications to obtaining diminishing swap and subsequence regret for downstream agents in online combinatorial optimization problems, as well as "transparent coverage" comes from Noarov et al. [2023]; they give other applications as well. The algorithm we give for online calibration has similar structure (randomizing between only two adjacent buckets) to a calibration algorithm first given by Foster and Hart [2021]. The multicalibration we give, based on exponential weights, is from Gupta et al. [2022].

Bibliography

Jacob Duncan Abernethy, Elad Hazan, and Alexander Rakhlin. An efficient algorithm for bandit linear optimization. In *Conference on Learning Theory*, 2008.

Dmitry Adamskiy, Wouter M Koolen, Alexey Chernov, and Vladimir Vovk. A closer look at adaptive regret. In *International Conference on Algorithmic Learning Theory*, pages 290–304. Springer, 2012.

Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of computing*, 8(1):121–164, 2012.

Avrim Blum and Thodoris Lykouris. Advancing subgroup fairness via sleeping experts. In *Innovations in Theoretical Computer Science Conference (ITCS)*, volume 11, 2020.

Avrim Blum and Yishay Mansour. From external to internal regret. *Journal of Machine Learning Research*, 8(6), 2007.

Dean P Foster and Sergiu Hart. Forecast hedging and calibration. *Journal of Political Economy*, 129(12):3447–3490, 2021.

Dean P Foster and Rakesh Vohra. Regret in the on-line decision problem. *Games and Economic Behavior*, 29(1-2):7–35, 1999.

Yoav Freund and Robert E Schapire. Game theory, on-line prediction and boosting. In *Proceedings of the ninth annual conference on Computational learning theory*, pages 325–332, 1996.

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

Varun Gupta, Christopher Jung, Georgy Noarov, Mallesh M Pai, and Aaron Roth. Online multivalid learning: Means, moments, and prediction intervals. In *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

Nika Haghtalab, Michael I Jordan, and Eric Zhao. A unifying perspective on multi-calibration: Unleashing game dynamics for multi-objective learning. *arXiv preprint arXiv:2302.10863*, 2023a.

Nika Haghtalab, Chara Podimata, and Kunhe Yang. Calibrated stackelberg games: Learning optimal commitments against calibrated agents. *arXiv preprint arXiv:2306.02704*, 2023b.

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

Elad Hazan and Comandur Seshadhri. Efficient learning algorithms for changing environments. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 393–400, 2009.

Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

Bobby Kleinberg, Renato Paes Leme, Jon Schneider, and Yifeng Teng. U-calibration: Forecasting for an unknown agent. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 5143–5145. PMLR, 2023.

Daniel Lee, Georgy Noarov, Mallesh Pai, and Aaron Roth. Online minimax multiobjective optimization: Multicalibeating and other applications. *Advances in Neural Information Processing Systems*, 35:29051–29063, 2022.

Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine learning*, 2:285–318, 1988.

Nick Littlestone and Manfred K Warmuth. The weighted majority algorithm. *Information and computation*, 108(2):212–261, 1994.

Georgy Noarov, Ramya Ramalingam, Aaron Roth, and Stephan Xie. High-dimensional prediction for sequential decision making. *arXiv preprint arXiv:2310.17651*, 2023.

RJ Williams. Sufficient conditions for nash equilibria in n-person games over reflexive banach spaces. *Journal of Optimization Theory and Applications*, 30:383–394, 1980.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, pages 928–936, 2003.