CHAPTER 4

Learning, Regret Minimization, and Equilibria

Avrim Blum and Yishay Mansour

Abstract

Many situations involve repeatedly making decisions in an uncertain environment: for instance, deciding what route to drive to work each day, or repeated play of a game against an opponent with an unknown strategy. In this chapter we describe learning algorithms with strong guarantees for settings of this type, along with connections to game-theoretic equilibria when all players in a system are simultaneously adapting in such a manner.

We begin by presenting algorithms for repeated play of a matrix game with the guarantee that against any opponent, they will perform nearly as well as the best fixed action in hindsight (also called the problem of *combining expert advice* or minimizing *external regret*). In a zero-sum game, such algorithms are guaranteed to approach or exceed the minimax value of the game, and even provide a simple proof of the minimax theorem. We then turn to algorithms that minimize an even stronger form of regret, known as *internal* or *swap* regret. We present a general reduction showing how to convert any algorithm for minimizing external regret to one that minimizes this stronger form of regret as well. Internal regret is important because when all players in a game minimize this stronger type of regret, the empirical distribution of play is known to converge to *correlated* equilibrium.

The third part of this chapter explains a different reduction: how to convert from the *full information* setting in which the action chosen by the opponent is revealed after each time step, to the *partial information* (bandit) setting, where at each time step only the payoff of the selected action is observed (such as in routing), and still maintain a small external regret.

Finally, we end by discussing routing games in the Wardrop model, where one can show that if all participants minimize their own external regret, then overall traffic is guaranteed to converge to an approximate Nash Equilibrium. This further motivates price-of-anarchy results.

4.1 Introduction

In this chapter we consider the problem of repeatedly making decisions in an uncertain environment. The basic setting is we have a space of N actions, such as what route to use to drive to work, or the rows of a matrix game like {rock, paper, scissors}. At each time step, the algorithm probabilistically chooses an action (say, selecting what route to take), the environment makes its “move” (setting the road congestions on that day), and the algorithm then incurs the loss for its action chosen (how long its route took). The process then repeats the next day. What we would like are adaptive algorithms that can perform well in such settings, as well as to understand the dynamics of the system when there are multiple players, all adjusting their behavior in such a way.

A key technique for analyzing problems of this sort is known as regret analysis. The motivation behind regret analysis can be viewed as the following: we design a sophisticated online algorithm that deals with various issues of uncertainty and decision making, and sell it to a client. Our algorithm runs for some time and incurs a certain loss. We would like to avoid the embarrassment that our client will come back to us and claim that in *retrospect* we could have incurred a much lower loss if we used his simple alternative policy π. The regret of our online algorithm is the difference between the loss of our algorithm and the loss using π.

Different notions of regret quantify differently what is considered to be a "simple" alternative policy. *External regret*, also called the problem of *combining expert advice*, compares performance to the best single action in retrospect. This implies that the simple alternative policy performs the same action in all time steps, which indeed is quite simple. Nonetheless, external regret provides a general methodology for developing online algorithms whose performance matches that of an optimal static offline algorithm by modeling the possible static solutions as different actions. In the context of machine learning, algorithms with good external regret bounds can be powerful tools for achieving performance comparable to the optimal prediction rule from some large class of hypotheses.

In Section 4.3 we describe several algorithms with particularly strong external regret bounds. We start with the very weak greedy algorithm, and build up to an algorithm whose loss is at most $O(\sqrt{T \log N})$ greater than that of the best action, where T is the number of time steps. That is, the regret per time step drops as $O(\sqrt{(\log N)/T})$. In Section 4.4 we show that in a zero-sum game, such algorithms are guaranteed to approach or exceed the value of the game, and even yield a simple proof of the minimax theorem.

A second category of alternative policies are those that consider the online sequence of actions and suggest a simple modification to it, such as "every time you bought IBM, you should have bought Microsoft instead." While one can study very general classes of modification rules, the most common form, known as *internal* or *swap* regret, allows one to modify the online action sequence by changing every occurrence of a given action i by an alternative action j. (The distinction between internal and swap regret is that internal regret allows only one action to be replaced by another, whereas swap regret allows any mapping from $\{1, \ldots, N\}$ to $\{1, \ldots, N\}$ and can be up to a factor N larger). In Section 4.5 we present a simple way to efficiently convert any external regret minimizing algorithm into one that minimizes swap regret with only a factor N increase in the regret term. Using the results for external regret this achieves a swap regret bound of $O(\sqrt{TN \log N})$. (Algorithms for swap regret have also been developed from first principles—see the Notes section of this chapter for references—but this procedure gives the best bounds known for efficient algorithms.)

The importance of swap regret is due to its tight connection to correlated equilibria, defined in Chapter 1. In fact, one way to think of a correlated equilibrium is that it is a distribution Q over the joint action space such that every player would have zero internal (or swap) regret when playing it. As we point out in Section 4.4, if each player can achieve swap regret ϵT, then the empirical distribution of the joint actions of the players will be an ϵ-correlated equilibrium.

We also describe how external regret results can be extended to the *partial information model*, also called the *multiarmed bandit* (MAB) problem. In this model, the online algorithm only gets to observe the loss of the action actually selected, and does not see the losses of the actions not chosen. For example, in the case of driving to work, you may only observe the travel time on the route you actually drive, and do not get to find out how long it would have taken had you chosen some alternative route. In Section 4.6 we present a general reduction, showing how to convert an algorithm with low external regret in the full information model to one for the partial information model (though the bounds produced are not the best known bounds for this problem).

Notice that the route-choosing problem can be viewed as a general-sum game: your travel time depends on the choices of the other drivers as well. In Section 4.7 we discuss results showing that in the Wardrop model of infinitesimal agents (considered in Chapter 18), if each driver acts to minimize external regret, then traffic flow over time can be shown to approach an approximate Nash equilibrium. This serves to further motivate *price-of-anarchy* results in this context, since it means they apply to the case that participants are using well-motivated self-interested *adaptive* behavior.

We remark that the results we present in this chapter are not always the strongest known, and the interested reader is referred to the recent book (Cesa-Bianchi and Lugosi, 2006) that gives a thorough coverage of many of the the topics in this chapter. See also the Notes section for further references.

4.2 Model and Preliminaries

We assume an adversarial online model where there are N available actions $X = \{1, \ldots, N\}$. At each time step t, an online algorithm H selects a distribution p^t over the N actions. After that, the adversary selects a loss vector $\ell^t \in [0,1]^N$, where $\ell_i^t \in [0,1]$ is the loss of the i-th action at time t. In the *full information model*, the online algorithm H receives the loss vector ℓ^t and experiences a loss $\ell_H^t = \sum_{i=1}^N p_i^t \ell_i^t$. (This can be viewed as an expected loss when the online algorithm selects action $i \in X$ with probability p_i^t.) In the *partial information model*, the online algorithm receives $(\ell_{k^t}^t, k^t)$, where k^t is distributed according to p^t, and $\ell_H^t = \ell_{k^t}^t$ is its loss. The loss of the i-th action during the first T time steps is $L_i^T = \sum_{t=1}^T \ell_i^t$, and the loss of H is $L_H^T = \sum_{t=1}^T \ell_H^t$.

The aim for the *external regret* setting is to design an online algorithm that will be able to approach the performance of the best algorithm from a given class of algorithms $\mathcal{G}$; namely, to have a loss close to $L_{\mathcal{G},\min}^T = \min_{g \in \mathcal{G}} L_g^T$. Formally we would like to minimize the external regret $R_{\mathcal{G}} = L_H^T - L_{\mathcal{G},\min}^T$, and $\mathcal{G}$ is called the *comparison class*. The most studied comparison class $\mathcal{G}$ is the one that consists of all the single actions, i.e., $\mathcal{G} = X$. In this chapter we concentrate on this important comparison class, namely, we want the online algorithm's loss to be close to $L_{\min}^T = \min_i L_i^T$, and let the external regret be $R = L_H^T - L_{\min}^T$.

External regret uses a fixed comparison class $\mathcal{G}$, but one can also envision a comparison class that depends on the online algorithm's actions. We can consider modification rules that modify the actions selected by the online algorithm, producing an alternative strategy which we will want to compete against. A *modification rule* F has as input the history and the current action selected by the online procedure and outputs a (possibly different) action. (We denote by F^t the function F at time t, including any dependency on the history.) Given a sequence of probability distributions p^t used by an online algorithm H, and a modification rule F, we define a new sequence of probability distributions $f^t = F^t(p^t)$, where $f_i^t = \sum_{j:F^t(j)=i} p_j^t$. The loss of the modified sequence is $L_{H,F} = \sum_t \sum_i f_i^t \ell_i^t$. Note that at time t the modification rule F shifts the probability that H assigned to action j to action $F^t(j)$. This implies that the modification rule F generates a different distribution, as a function of the online algorithm's distribution p^t.

We will focus on the case of a finite set $\mathcal{F}$ of memoryless modification rules (they do not depend on history). Given a sequence of loss vectors, the regret of an online algorithm H with respect to the modification rules $\mathcal{F}$ is

$$R_{\mathcal{F}} = \max_{F\in\mathcal{F}} \left\{L_H^T - L_{H,F}^T\right\}.$$

Note that the external regret setting is equivalent to having a set $\mathcal{F}^{\text{ex}}$ of N modification rules F_i, where F_i always outputs action i. For *internal regret*, the set $\mathcal{F}^{\text{in}}$ consists of $N(N-1)$ modification rules $F_{i,j}$, where $F_{i,j}(i) = j$ and $F_{i,j}(i') = i'$ for $i' \neq i$. That is, the internal regret of H is

$$\max_{F\in\mathcal{F}^{\text{in}}} \left\{L_H^T - L_{H,F}^T\right\} = \max_{i,j\in X} \left\{\sum_{t=1}^{T} p_i^t\left(\ell_i^t - \ell_j^t\right)\right\}.$$

A more general class of memoryless modification rules is *swap regret* defined by the class $\mathcal{F}^{\text{sw}}$, which includes all N^N functions $F : \{1,\dots,N\} \to \{1,\dots,N\}$, where the function F swaps the current online action i with $F(i)$ (which can be the same or a different action). That is, the swap regret of H is

$$\max_{F\in\mathcal{F}^{\text{sw}}} \left\{L_H^T - L_{H,F}^T\right\} = \sum_{i=1}^{N} \max_{j\in X} \left\{\sum_{t=1}^{T} p_i^t\left(\ell_i^t - \ell_j^t\right)\right\}.$$

Note that since $\mathcal{F}^{\text{ex}} \subseteq \mathcal{F}^{\text{sw}}$ and $\mathcal{F}^{\text{in}} \subseteq \mathcal{F}^{\text{sw}}$, both external and internal regret are upper-bounded by swap regret. (See also Exercises 4.1 and 4.2.)

4.3 External Regret Minimization

Before describing the external regret results, we begin by pointing out that it is not possible to guarantee low regret with respect to the overall optimal *sequence* of decisions in hindsight, as is done in competitive analysis (Borodin and El-Yaniv, 1998; Sleator and Tarjan, 1985). This will motivate why we will be concentrating on more restricted comparison classes. In particular, let $\mathcal{G}_{\text{all}}$ be the set of all functions mapping times $\{1,\dots,T\}$ to actions $X = \{1,\dots,N\}$.

Theorem 4.1 *For any online algorithm H there exists a sequence of T loss vectors such that regret $R_{\mathcal{G}_{\text{all}}}$ is at least $T(1-1/N)$.*

PROOF The sequence is simply as follows: at each time t, the action i_t of lowest probability p_i^t gets a loss of 0, and all the other actions get a loss of 1. Since $\min_i\{p_i^t\} \leq 1/N$, this means the loss of H in T time steps is at least $T(1 - 1/N)$. On the other hand, there exists $g \in \mathcal{G}_{\text{all}}$, namely $g(t) = i_t$, with a total loss of 0. □

The above proof shows that if we consider all possible functions, we have a very large regret. For the rest of the section we will use the comparison class $\mathcal{G}_a = \{g_i : i \in X\}$, where g_i always selects action i. Namely, we compare the online algorithm to the best single action.

4.3.1 Warmup: Greedy and Randomized-Greedy Algorithms

In this section, for simplicity we will assume that all losses are either 0 or 1 (rather than a real number in [0, 1]), which will simplify notation and proofs, although everything presented can be easily extended to the general case.

Our first attempt to develop a good regret minimization algorithm will be to consider the greedy algorithm. Recall that $L_i^t = \sum_{\tau=1}^{t} \ell_i^\tau$, namely the cumulative loss up to time t of action i. The `Greedy` algorithm at each time t selects action $x^t = \arg\min_{i \in X} L_i^{t-1}$ (if there are multiple actions with the same cumulative loss, it prefers the action with the lowest index). Formally:

```
Greedy Algorithm
Initially:   $x^1 = 1$.
At time $t$: Let $L_{\min}^{t-1} = \min_{i \in X} L_i^{t-1}$, and $S^{t-1} = \{i : L_i^{t-1} = L_{\min}^{t-1}\}$.
             Let $x^t = \min S^{t-1}$.
```

Theorem 4.2 *The* `Greedy` *algorithm, for any sequence of losses has*

$$L_{\texttt{Greedy}}^T \leq N \cdot L_{\min}^T + (N - 1).$$

PROOF At each time t such that `Greedy` incurs a loss of 1 and $L_{\min}^t$ does not increase, at least one action is removed from S^t. This can occur at most N times before $L_{\min}^t$ increases by 1. Therefore, `Greedy` incurs loss at most N between successive increments in $L_{\min}^t$. More formally, this shows inductively that $L_{\texttt{Greedy}}^t \leq N - |S^t| + N \cdot L_{\min}^t$. □

The above guarantee on `Greedy` is quite weak, stating only that its loss is at most a factor of N larger than the loss of the best action. The following theorem shows that this weakness is shared by any deterministic online algorithm. (A deterministic algorithm concentrates its entire weight on a single action at each time step.)

Theorem 4.3 *For any deterministic algorithm D there exists a loss sequence for which $L_D^T = T$ and $L_{\min}^T = \lfloor T/N \rfloor$.*

Note that the above theorem implies that $L_D^T \geq N \cdot L_{\min}^T + (T \bmod N)$, which almost matches the upper bound for `Greedy` (Theorem 4.2).

PROOF Fix a deterministic online algorithm D and let x^t be the action it selects at time t. We will generate the loss sequence in the following way. At time t, let the loss of x^t be 1 and the loss of any other action be 0. This ensures that D incurs loss 1 at each time step, so $L_D^T = T$.

Since there are N different actions, there is some action that algorithm D has selected at most $\lfloor T/N \rfloor$ times. By construction, only the actions selected by D ever have a loss, so this implies that $L_{\min}^T \leq \lfloor T/N \rfloor$. □

Theorem 4.3 motivates considering randomized algorithms. In particular, one weakness of the greedy algorithm was that it had a deterministic *tie breaker*. One can hope that if the online algorithm splits its weight between all the currently best actions, better performance could be achieved. Specifically, let `Randomized Greedy` (RG) be the procedure that assigns a uniform distribution over all those actions with minimum total loss so far. We now will show that this algorithm achieves a significant performance improvement: its loss is at most an $O(\log N)$ factor from the best action, rather than $O(N)$. (This is similar to the analysis of the randomized marking algorithm in competitive analysis.)

```
Randomized Greedy (RG) Algorithm
Initially:   $p_i^1 = 1/N$ for $i \in X$.
At time $t$:   Let $L_{\min}^{t-1} = \min_{i \in X} L_i^{t-1}$, and $S^{t-1} = \{i : L_i^{t-1} = L_{\min}^{t-1}\}$.
             Let $p_i^t = 1/|S^{t-1}|$ for $i \in S^{t-1}$ and $p_i^t = 0$ otherwise.
```

Theorem 4.4 *The* `Randomized Greedy (RG)` *algorithm, for any loss sequence, has*

$$L_{\text{RG}}^T \leq (\ln N) + (1 + \ln N) L_{\min}^T .$$

PROOF The proof follows from showing that the loss incurred by `Randomized Greedy` between successive increases in $L_{\min}^t$ is at most $1 + \ln N$. Specifically, let t_j denote the time step at which $L_{\min}^t$ first reaches a loss of j, so we are interested in the loss of `Randomized Greedy` between time steps t_j and t_{j+1}. At time any t we have $1 \leq |S^t| \leq N$. Furthermore, if at time $t \in (t_j, t_{j+1}]$ the size of S^t shrinks by k from some size n' down to $n' - k$, then the loss of the online algorithm `RG` is k/n', since each such action has weight $1/n'$. Finally, notice that we can upper bound k/n' by $1/n' + 1/(n'-1) + \cdots + 1/(n'-k+1)$. Therefore, over the entire time-interval $(t_j, t_{j+1}]$, the loss of `Randomized Greedy` is at most:

$$1/N + 1/(N-1) + 1/(N-2) + \cdots + 1/1 \leq 1 + \ln N.$$

More formally, this shows inductively that $L_{\text{RG}}^t \leq (1/N + 1/(N-1) + \cdots + 1/(|S^t| + 1)) + (1 + \ln N) \cdot L_{\min}^t$. □

4.3.2 Randomized Weighted Majority Algorithm

Although Randomized Greedy achieved a significant performance gain compared to the Greedy algorithm, we still have a logarithmic ratio to the best action. Looking more closely at the proof, one can see that the losses are greatest when the sets S^t are small, since the online loss can be viewed as proportional to $1/|S^t|$. One way to overcome this weakness is to give some weight to actions which are currently "near best." That is, we would like the probability mass on some action to decay gracefully with its distance to optimality. This is the idea of the *Randomized Weighted Majority* algorithm of Littlestone and Warmuth.

Specifically, in the Randomized Weighted Majority algorithm, we give an action i whose total loss so far is L_i a *weight* $w_i = (1-\eta)^{L_i}$, and then choose probabilities proportional to the weights: $p_i = w_i / \sum_{j=1}^{N} w_j$. The parameter η will be set to optimize certain trade-offs but conceptually think of it as a small constant, say 0.01. In this section we will again assume losses in $\{0, 1\}$ rather than $[0, 1]$ because it allows for an especially intuitive interpretation of the proof (Theorem 4.5). We then relax this assumption in the next section (Theorem 4.6).

```
Randomized Weighted Majority (RWM) Algorithm
Initially:   $w_i^1 = 1$ and $p_i^1 = 1/N$, for $i \in X$.
At time $t$:   If $\ell_i^{t-1} = 1$, let $w_i^t = w_i^{t-1}(1-\eta)$; else ($\ell_i^{t-1} = 0$) let $w_i^t = w_i^{t-1}$.
             Let $p_i^t = w_i^t / W^t$, where $W^t = \sum_{i \in X} w_i^t$.
```

Algorithm RWM and Theorem 4.5 can be generalized to losses in $[0, 1]$ by replacing the update rule with $w_i^t = w_i^{t-1}(1-\eta)^{\ell_i^{t-1}}$ (see Exercise 4.3).

Theorem 4.5 *For $\eta \le 1/2$, the loss of* Randomized Weighted Majority (RWM) *on any sequence of binary* $\{0, 1\}$ *losses satisfies*

$$L_{\mathtt{RWM}}^T \le (1+\eta)L_{\min}^T + \frac{\ln N}{\eta}.$$

Setting $\eta = \min\{\sqrt{(\ln N)/T}, 1/2\}$ *yields* $L_{\mathtt{RWM}}^T \le L_{\min}^T + 2\sqrt{T \ln N}$.

(Note: The second part of the theorem assumes T is known in advance. If T is unknown, then a "guess and double" approach can be used to set η with just a constant-factor loss in regret. In fact, one can achieve the potentially better bound $L_{\mathtt{RWM}}^T \le L_{\min}^T + 2\sqrt{L_{\min} \ln N}$ by setting $\eta = \min\{\sqrt{(\ln N)/L_{\min}}, 1/2\}$.)

PROOF The key to the proof is to consider the total weight W^t. What we will show is that anytime the online algorithm has significant expected loss, the total weight must drop substantially. We will then combine this with the fact that $W^{T+1} \ge \max_i w_i^{T+1} = (1-\eta)^{L_{\min}^T}$ to achieve the desired bound.

Specifically, let $F^t = (\sum_{i:\ell_i^t=1} w_i^t)/W^t$ denote the fraction of the weight W^t that is on actions that experience a loss of 1 at time t; so, F^t equals the expected loss of algorithm RWM at time t. Now, each of the actions experiencing a loss of 1 has its weight multiplied by $(1-\eta)$ while the rest are unchanged. Therefore, $W^{t+1} = W^t - \eta F^t W^t = W^t(1-\eta F^t)$. In other words, the proportion of the weight removed from the system at each time t is exactly proportional to the expected loss of the online algorithm. Now, using the fact that $W^1 = N$ and using our lower bound on W^{T+1} we have

$$(1-\eta)^{L^T_{\min}} \leq W^{T+1} = W^1 \prod_{t=1}^{T}(1-\eta F^t) = N\prod_{t=1}^{T}(1-\eta F^t).$$

Taking logarithms,

$$\begin{aligned} L^T_{\min}\ln(1-\eta) &\leq (\ln N) + \sum_{t=1}^{T}\ln(1-\eta F^t) \\ &\leq (\ln N) - \sum_{t=1}^{T}\eta F^t \\ &\quad \text{(Using the inequality } \ln(1-z) \leq -z\text{)} \\ &= (\ln N) - \eta L^T_{\mathtt{RWM}} \\ &\quad \text{(by definition of } F^t\text{)} \end{aligned}$$

Therefore,

$$\begin{aligned} L^T_{\mathtt{RWM}} &\leq \frac{-L^T_{\min}\ln(1-\eta)}{\eta} + \frac{\ln(N)}{\eta} \\ &\leq (1+\eta)L^T_{\min} + \frac{\ln(N)}{\eta}, \\ &\quad \text{(Using the inequality } -\ln(1-z) \leq z + z^2 \text{ for } 0 \leq z \leq \tfrac{1}{2}\text{)} \end{aligned}$$

which completes the proof. □

4.3.3 Polynomial Weights Algorithm

The Polynomial Weights (PW) algorithm is a natural extension of the RWM algorithm to losses in [0, 1] (or even to the case of both losses and gains, see Exercise 4.4) that maintains the same proof structure as that used for RWM and in addition performs especially well in the case of small losses.

```
Polynomial Weights (PW) Algorithm
Initially:   $w_i^1 = 1$ and $p_i^1 = 1/N$, for $i \in X$.
At time $t$:   Let $w_i^t = w_i^{t-1}(1 - \eta\ell_i^{t-1})$.
             Let $p_i^t = w_i^t / W^t$, where $W^t = \sum_{i\in X} w_i^t$.
```

Notice that the only difference between PW and RWM is in the update step. In particular, it is no longer necessarily the case that an action of total loss L has weight $(1-\eta)^L$. However, what *is* maintained is the property that if the algorithm's loss at time t is F^t, then exactly an ηF^t fraction of the total weight is removed from the system. Specifically, from the update rule we have $W^{t+1} = W^t - \sum_i \eta w_i^t \ell_i^t = W^t(1-\eta F^t)$ where $F^t = (\sum_i w_i^t \ell_i^t)/W^t$ is the loss of PW at time t. We can use this fact to prove the following.

Theorem 4.6 *The* Polynomial Weights (PW) *algorithm, using* $\eta \leq 1/2$*, for any* $[0,1]$*-valued loss sequence and for any* k *has,*

$$L_{\texttt{PW}}^T \leq L_k^T + \eta Q_k^T + \frac{\ln(N)}{\eta},$$

where $Q_k^T = \sum_{t=1}^T (\ell_k^t)^2$. *Setting* $\eta = \min\{\sqrt{(\ln N)/T}, 1/2\}$ *and noting that* $Q_k^T \leq T$*, we have* $L_{\texttt{PW}}^T \leq L_{\min}^T + 2\sqrt{T \ln N}$.[1]

PROOF As noted above, we have $W^{t+1} = W^t(1 - \eta F^t)$, where F^t is PW's loss at time t. So, as with the analysis of RWM, we have $W^{T+1} = N \prod_{t=1}^T (1 - \eta F^t)$ and therefore

$$\ln W^{T+1} \;=\; \ln N + \sum_{t=1}^{T} \ln(1 - \eta F^t) \;\leq\; \ln N - \eta \sum_{t=1}^{T} F^t \;=\; \ln N - \eta L_{\texttt{PW}}^T.$$

Now for the lower bound, we have

$$\begin{aligned}
\ln W^{T+1} &\geq \ln w_k^{T+1} \\
&= \sum_{t=1}^{T} \ln\left(1 - \eta \ell_k^t\right) \\
&\quad \text{(using the recursive definition of weights)} \\
&\geq -\sum_{t=1}^{T} \eta \ell_k^t - \sum_{t=1}^{T} \left(\eta \ell_k^t\right)^2 \\
&\quad \text{(using the inequality } \ln(1-z) \geq -z - z^2 \text{ for } 0 \leq z \leq \tfrac{1}{2}) \\
&= -\eta L_k^T - \eta^2 Q_k^T.
\end{aligned}$$

Combining the upper and lower bounds on $\ln W^{T+1}$ we have:

$$-\eta L_k^T - \eta^2 Q_k^T \leq \ln N - \eta L_{\texttt{PW}}^T,$$

which yields the theorem. □

4.3.4 Lower Bounds

An obvious question is whether one can significantly improve the bound in Theorem 4.6. We will show two simple results that imply that the regret bound is near optimal (see Exercise 4.5 for a better lower bound). The first result shows that one cannot hope to get sublinear regret when T is small compared to $\log N$, and the second shows that one cannot hope to achieve regret $o(\sqrt{T})$ even when $N = 2$.

Theorem 4.7 *Consider* $T < \log_2 N$. *There exists a stochastic generation of losses such that, for any online algorithm* $R1$*, we have* $E[L_{R1}^T] = T/2$ *and yet* $L_{\min}^T = 0$.

[1] Again, for simplicity we assume that the number of time steps T is given as a parameter to the algorithm; otherwise, one can use a "guess and double" method to set η.

PROOF Consider the following sequence of losses. At time $t = 1$, a random subset of $N/2$ actions gets a loss of 0 and the rest gets a loss of 1. At time $t = 2$, a random subset of $N/4$ of the actions that had loss 0 at time $t = 1$ gets a loss of 0, and the rest (including actions that had a loss of 1 at time 1) gets a loss of 1. This process repeats: at each time step, a random subset of half of the actions that have received loss 0 so far gets a loss of 0, while all the rest gets a loss of 1. Any online algorithm incurs an expected loss of 1/2 at each time step, because at each time step t the expected fraction of probability mass p_i^t on actions that receive a loss of 0 is at most 1/2. Yet, for $T < \log_2 N$ there will always be some action with total loss of 0. □

Theorem 4.8 *Consider $N = 2$. There exists a stochastic generation of losses such that, for any online algorithm R2, we have $E[L_{R2}^T - L_{\min}^T] = \Omega(\sqrt{T})$.*

PROOF At time t, we flip a fair coin and set $\ell^t = z_1 = (0, 1)$ with probability 1/2 and $\ell^t = z_2 = (1, 0)$ with probability 1/2. For any distribution p^t the expected loss at time t is exactly 1/2. Therefore any online algorithm $R2$ has expected loss of $T/2$.

Given a sequence of T such losses, with $T/2 + y$ losses z_1 and $T/2 - y$ losses z_2, we have $T/2 - L_{\min}^T = |y|$. It remains to lower bound $E[|y|]$. Note that the probability of y is $\binom{T}{T/2+y}/2^T$, which is upper bounded by $O(1/\sqrt{T})$ (using a Sterling approximation). This implies that with a constant probability we have $|y| = \Omega(\sqrt{T})$, which completes the proof. □

4.4 Regret Minimization and Game Theory

In this section we outline the connection between regret minimization and central concepts in game theory. We start by showing that in a two-player constant sum game, a player with external regret sublinear in T will have an average payoff that is at least the value of the game, minus a vanishing error term. For a general game, we will see that if all the players use procedures with sublinear *swap-regret*, then they will converge to an approximate *correlated* equilibrium. We also show that for a player who minimizes swap-regret, the frequency of playing dominated actions is vanishing.

4.4.1 Game Theoretic Model

We start with the standard definitions of a game (see also Chapter 1). A game $G = \langle M, (X_i), (s_i) \rangle$ has a finite set M of m players. Player i has a set X_i of N actions and a loss function $s_i : X_i \times (\times_{j \neq i} X_j) \rightarrow [0, 1]$ that maps the action of player i and the actions of the other players to a real number. (We have scaled losses to [0, 1].) The joint action space is $X = \times X_i$.

We consider a player i that plays a game G for T time steps using an online procedure ON. At time step t, player i plays a distribution (mixed action) P_i^t, while the other players play the joint distribution P_{-i}^t. We denote by $\ell_{\mathtt{ON}}^t$ the loss of player i at time t, i.e.,

$E_{x \sim P^t}[s_i(x^t)]$, and its cumulative loss is $L_{\mathtt{ON}}^T = \sum_{t=1}^T \ell_{\mathtt{ON}}^t$.[2] It is natural to define, for player i at time t, the loss vector as $\ell^t = (\ell_1^t, \ldots, \ell_N^t)$, where $\ell_j^t = E_{x_{-i}^t \sim P_{-i}^t}[s_i(x_j^t, x_{-i}^t)]$. Namely, ℓ_j^t is the loss player i would have observed if at time t it had played action x_j. The cumulative loss of action $x_j \in X_i$ of player i is $L_j^T = \sum_{t=1}^T \ell_j^t$, and $L_{\min}^T = \min_j L_j^T$.

4.4.2 Constant Sum Games and External Regret Minimization

A two-player constant sum game $G = \langle \{1, 2\}, (X_i), (s_i) \rangle$ has the property that for some constant c, for every $x_1 \in X_1$ and $x_2 \in X_2$ we have $s_1(x_1, x_2) + s_2(x_1, x_2) = c$. It is well known that any constant sum game has a well-defined *value* (v_1, v_2) for the game, and player $i \in \{1, 2\}$ has a mixed strategy which guarantees that its expected loss is at most v_i, regardless of the other player's strategy. (See Owen, 1982, for more details.) In such games, external regret-minimization procedures provide the following guarantee.

Theorem 4.9 *Let G be a constant sum game with game value (v_1, v_2). If player $i \in \{1, 2\}$ plays for T steps using a procedure* ON *with external regret R, then its average loss $\frac{1}{T} L_{\mathtt{ON}}^T$ is at most $v_i + R/T$.*

PROOF Let q be the mixed strategy corresponding to the observed frequencies of the actions player 2 has played; that is, $q_j = \sum_{t=1}^T P_{2,j}^t / T$, where $P_{2,j}^t$ is the weight player 2 gives to action j at time t. By the theory of constant sum games, for any mixed strategy q of player 2, player 1 has some action $x_k \in X_1$ such that $E_{x_2 \sim q}[s_1(x_k, x_2)] \leq v_1$ (see Owen, 1982). This implies, in our setting, that if player 1 has always played action x_k, then its loss would be at most $v_1 T$. Therefore $L_{\min}^T \leq L_k^T \leq v_1 T$. Now, using the fact that player 1 is playing a procedure ON with external regret R, we have that $L_{\mathtt{ON}}^T \leq L_{\min}^T + R \leq v_1 T + R$. □

Thus, using a procedure with regret $R = O(\sqrt{T \log N})$ as in Theorem 4.6 will guarantee average loss at most $v_i + O(\sqrt{(\log N)/T})$.

In fact, we can use the existence of external regret minimization algorithms to prove the minimax theorem of two-player zero-sum games. For player 1, let $v_{\min}^1 = \min_{x_1 \in X_1} \max_{z \in \Delta(X_2)} E_{x_2 \sim z}[s_1(x_1, x_2)]$ and $v_{\max}^1 = \max_{x_2 \in X_2} \min_{z \in \Delta(X_1)} E_{x_1 \sim z}[s_1(x_1, x_2)]$. That is, $v_{\min}^1$ is the best loss that player 1 can guarantee for itself if it is told the mixed action of player 2 in advance. Similarly, $v_{\max}^1$ is the best loss that player 1 can guarantee to itself if it has to go first in selecting a mixed action, and player 2's action may then depend on it. The minimax theorem states that $v_{\min}^1 = v_{\max}^1$. Since $s_1(x_1, x_2) = -s_2(x_1, x_2)$ we can similarly define $v_{\min}^2 = -v_{\max}^1$ and $v_{\max}^2 = -v_{\min}^1$.

In the following we give a proof of the minimax theorem based on the existence of external regret algorithms. Assume for contradiction that $v_{\max}^1 = v_{\min}^1 + \gamma$ for some $\gamma > 0$ (it is easy to see that $v_{\max}^1 \geq v_{\min}^1$). Consider both players playing a regret

[2] Alternatively, we could consider x_i^t as a random variable distributed according to P_i^t, and similarly discuss the expected loss. We prefer the above presentation for consistency with the rest of the chapter.

minimization algorithm for T steps having external regret of at most R, such that $R/T < \gamma/2$. Let L_{ON} be the loss of player 1 and note that $-L_{\mathrm{ON}}$ is the loss of player 2. Let $L^i_{\min}$ be the cumulative loss of the best action of player $i \in \{1, 2\}$. As before, let q_i be the mixed strategy corresponding to the observed frequencies of actions of player $i \in \{1, 2\}$. Then, $L^1_{\min}/T \leq v^1_{\min}$, since for $L^1_{\min}$ we select the best action with respect to a specific mixed action, namely q_2. Similarly, $L^2_{\min}/T \leq v^2_{\min}$. The regret minimization algorithms guarantee for player 1 that $L_{\mathrm{ON}} \leq L^1_{\min} + R$, and for player 2 that $-L_{\mathrm{ON}} \leq L^2_{\min} + R$. Combining the inequalities we have:

$$T v^1_{\max} - R = -T v^2_{\max} - R \leq -L^2_{\min} - R \leq L_{\mathrm{ON}} \leq L^1_{\min} + R \leq T v^1_{\min} + R.$$

This implies that $v^1_{\max} - v^1_{\min} \leq 2R/T < \gamma$, which is a contradiction. Therefore, $v^1_{\max} = v^1_{\min}$, which establishes the minimax theorem.

4.4.3 Correlated Equilibrium and Swap Regret Minimization

We first define the relevant modification rules and establish the connection between them and equilibrium notions. For $x_1, b_1, b_2 \in X_i$, let $\mathrm{switch}_i(x_1, b_1, b_2)$ be the following modification function of the action x_1 of player i:

$$\mathrm{switch}_i(x_1, b_1, b_2) = \begin{cases} b_2 & \text{if } x_1 = b_1 \\ x_1 & \text{otherwise} \end{cases}$$

Given a modification function f for player i, we can measure the regret of player i with respect to f as the decrease in its loss, i.e.,

$$\mathrm{regret}_i(x, f) = s_i(x) - s_i(f(x_i), x_{-i}).$$

For example, when we consider $f(x_1) = \mathrm{switch}_i(x_1, b_1, b_2)$, for a fixed $b_1, b_2 \in X_i$, then $\mathrm{regret}_i(x, f)$ is measuring the regret player i has for playing action b_1 rather than b_2, when the other players play x_{-i}.

A *correlated equilibrium* is a distribution P over the joint action space with the following property. Imagine a correlating device draws a vector of actions $x \in X$ using distribution P over X, and gives player i the action x_i from x. (Player i is not given any other information regarding x.) The probability distribution P is a correlated equilibrium if, for each player, it is a best response to play the suggested action, provided that the other players also do not deviate. (For a more detailed discussion of correlated equilibrium, see Chapter 1.)

Definition 4.10 A joint probability distribution P over X is a *correlated equilibrium* if for every player i, and any actions $b_1, b_2 \in X_i$, we have that

$$E_{x \sim P}[\mathrm{regret}_i(x, \mathrm{switch}_i(\cdot, b_1, b_2))] \leq 0.$$

An equivalent definition that extends more naturally to the case of approximate equilibria is to say that rather than only switching between a pair of actions, we allow simultaneously replacing every action in X_i with another action in X_i (possibly the same action). A distribution P is a correlated equilibrium iff for any function $F : X_i \to X_i$ we have $E_{x \sim P}[\mathrm{regret}_i(x, F)] \leq 0$.

We now define an ϵ-correlated equilibrium. An ϵ-correlated equilibrium is a distribution P such that each player has in expectation at most an ϵ incentive to deviate. Formally,

Definition 4.11 A joint probability distribution P over X is an ϵ-correlated equilibria if for every player i and for any function $F_i : X_i \rightarrow X_i$, we have $E_{x \sim P}[\text{regret}_i(x, F_i)] \leq \epsilon$.

The following theorem relates the empirical distribution of the actions performed by each player, their swap regret, and the distance to correlated equilibrium.

Theorem 4.12 *Let $G = \langle M, (X_i), (s_i) \rangle$ be a game and assume that for T time steps every player follows a strategy that has swap regret of at most R. Then, the empirical distribution Q of the joint actions played by the players is an (R/T)-correlated equilibrium.*

PROOF The empirical distribution Q assigns to every P^t a probability of $1/T$. Fix a function $F : X_i \rightarrow X_i$ for player i. Since player i has swap regret at most R, we have $L^T_{\mathtt{ON}} \leq L^T_{\mathtt{ON},F} + R$, where $L^T_{\mathtt{ON}}$ is the loss of player i. By definition of the regret function, we therefore have

$$L^T_{\mathtt{ON}} - L^T_{\mathtt{ON},F} = \sum_{t=1}^{T} E_{x^t \sim P^t}[s_i(x^t)] - \sum_{t=1}^{T} E_{x^t \sim P^t}\left[s_i\left(F\left(x^t_i\right), x^t_{-i}\right)\right]$$

$$= \sum_{t=1}^{T} E_{x^t \sim P^t}[\text{regret}_i(x^t, F)] = T \cdot E_{x \sim Q}[\text{regret}_i(x, F)].$$

Therefore, for any function $F_i : X_i \rightarrow X_i$ we have $E_{x \sim Q}[\text{regret}_i(x, F_i)] \leq R/T$. □

The above theorem states that the payoff of each player is its payoff in some approximate correlated equilibrium. In addition, it relates the swap regret to the distance from equilibrium. Note that if the average swap regret vanishes then the procedure converges, in the limit, to the set of correlated equilibria.

4.4.4 Dominated Strategies

We say that an action $x_j \in X_i$ is *ϵ-dominated* by action $x_k \in X_i$ if for any $x_{-i} \in X_{-i}$ we have $s_i(x_j, x_{-i}) \geq \epsilon + s_i(x_k, x_{-i})$. Similarly, action $x_j \in X_i$ is *ϵ-dominated* by a mixed action $y \in \Delta(X_i)$ if for any $x_{-i} \in X_{-i}$ we have $s_i(x_j, x_{-i}) \geq \epsilon + E_{x_d \sim y}[s_i(x_d, x_{-i})]$.

Intuitively, a good learning algorithm ought to be able to learn not to play actions that are ϵ-dominated by others, and in this section we show that indeed if player i plays a procedure with sublinear swap regret, then it will very rarely play dominated actions. More precisely, let action x_j be ϵ-dominated by action $x_k \in X_i$. Using our notation, this implies that for any x_{-i} we have that $\text{regret}_i(x, \text{switch}_i(\cdot, x_j, x_k)) \geq \epsilon$. Let D_ϵ be the set of ϵ-dominated actions of player i, and let w be the weight that player i puts on actions in D_ϵ, averaged over time, i.e., $w = \frac{1}{T}\sum_{t=1}^{T}\sum_{j\in D_\epsilon} P^t_{i,j}$. Player i's swap regret is at least $\epsilon w T$ (since we could replace each action in D_ϵ with the action that dominates it). So, if the player's swap regret is R, then $\epsilon w T \leq R$. Therefore, the time-average weight that player i puts on the set of ϵ-dominated actions is at most $R/(\epsilon T)$, which tends to 0 if R is sublinear in T. That is:

Theorem 4.13 *Consider a game G and a player i that uses a procedure of swap regret R for T time steps. Then the average weight that player i puts on the set of ϵ-dominated actions is at most $R/(\epsilon T)$.*

We remark that in general the property of having low *external* regret is not sufficient by itself to give such a guarantee, though the algorithms RWM and PW do indeed have such a guarantee (see Exercise 4.8).

4.5 Generic Reduction from External to Swap Regret

In this section we give a black-box reduction showing how any procedure A achieving good external regret can be used as a subroutine to achieve good swap regret as well. The high-level idea is as follows (see also Figure 4.1). We will instantiate N copies $A_1, \ldots, A_N$ of the external-regret procedure. At each time step, these procedures will each give us a probability vector, which we will combine in a particular way to produce our own probability vector p. When we receive a loss vector ℓ, we will partition it among the N procedures, giving procedure A_i a fraction p_i (p_i is our probability mass on action i), so that A_i's belief about the loss of action j is $\sum_t p_i^t \ell_j^t$, and matches the cost we would incur putting i's probability mass on j. In the proof, procedure A_i will, in some sense, be responsible for ensuring low regret of the $i \to j$ variety. The key to making this work is that we will be able to define the p's so that the sum of the losses of the procedures A_i on their own loss vectors matches our overall true loss. Recall the definition of an R external regret procedure.

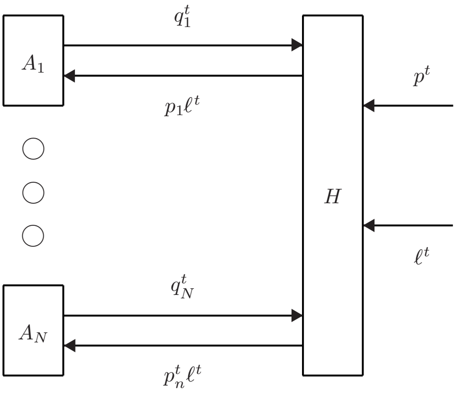

Figure 4.1. The structure of the swap regret reduction.

Definition 4.14 An R external regret procedure A guarantees that for any sequence of T losses ℓ^t and for any action $j \in \{1, \ldots, N\}$, we have

$$L_A^T = \sum_{t=1}^{T} \ell_A^t \leq \sum_{t=1}^{T} \ell_j^t + R = L_j^T + R.$$

We assume we have N copies $A_1, \ldots, A_N$ of an R external regret procedure. We combine the N procedures to one master procedure H as follows. At each time step t, each procedure A_i outputs a distribution q_i^t, where $q_{i,j}^t$ is the fraction it assigns action j. We compute a single distribution p^t such that $p_j^t = \sum_i p_i^t q_{i,j}^t$. That is, $p^t = p^t Q^t$, where p^t is our distribution and Q^t is the matrix of $q_{i,j}^t$. (We can view p^t as a stationary distribution of the Markov Process defined by Q^t, and it is well known that such a p^t exists and is efficiently computable.) For intuition into this choice of p^t, notice that it implies we can consider action selection in two equivalent ways. The first is simply using the distribution p^t to select action j with probability p_j^t. The second is to select procedure A_i with probability p_i^t and then to use A_i to select the action (which produces distribution $p^t Q^t$).

When the adversary returns the loss vector ℓ^t, we return to each A_i the loss vector $p_i \ell^t$. So, procedure A_i experiences loss $(p_i^t \ell^t) \cdot q_i^t = p_i^t(q_i^t \cdot \ell^t)$.

Since A_i is an R external regret procedure, for any action j, we have,

$$\sum_{t=1}^{T} p_i^t \left(q_i^t \cdot \ell^t\right) \leq \sum_{t=1}^{T} p_i^t \ell_j^t + R \tag{4.1}$$

If we sum the losses of the N procedures at a given time t, we get $\sum_i p_i^t(q_i^t \cdot \ell^t) = p^t Q^t \ell^t$, where p^t is the row vector of our distribution, Q^t is the matrix of $q_{i,j}^t$, and ℓ^t is viewed as a column vector. By design of p^t, we have $p^t Q^t = p^t$. So, the sum of the perceived losses of the N procedures is equal to our actual loss $p^t \ell^t$.

Therefore, summing equation (4.1) over all N procedures, the left-hand side sums to L_H^T, where H is our master online procedure. Since the right-hand side of equation (4.1) holds for any j, we have that for any function $F : \{1, \ldots, N\} \to \{1, \ldots, N\}$,

$$L_H^T \leq \sum_{i=1}^{N} \sum_{t=1}^{T} p_i^t \ell_{F(i)}^t + NR = L_{H,F}^T + NR$$

Therefore we have proven the following theorem.

Theorem 4.15 *Given an R external regret procedure, the master online procedure H has the following guarantee. For every function $F : \{1, \ldots, N\} \to \{1, \ldots, N\}$,*

$$L_H \leq L_{H,F} + NR,$$

i.e., the swap regret of H is at most NR.

Using Theorem 4.6, we can immediately derive the following corollary.

Corollary 4.16 *There exists an online algorithm H such that for every function $F : \{1, \ldots, N\} \to \{1, \ldots, N\}$, we have that*

$$L_H \leq L_{H,F} + O(N\sqrt{T \log N}),$$

i.e., the swap regret of H is at most $O(N\sqrt{T \log N})$.

Remark. See Exercise 4.6 for an improvement to $O(\sqrt{NT \log N})$.

4.6 The Partial Information Model

In this section we show, for external regret, a simple reduction from the partial information to the full information model.[3] The main difference between the two models is that in the full information model, the online procedure has access to the loss of every action. In the partial information model the online procedure receives as feedback only the loss of a single action, the action it performed. This very naturally leads to an *exploration versus exploitation trade-off* in the partial information model, and essentially any online procedure will have to somehow *explore* the various actions and estimate their loss.

The high-level idea of the reduction is as follows. Assume that the number of time steps T is given as a parameter. We will partition the T time steps into K blocks. The procedure will use the same distribution over actions in all the time steps of any given block, except it will also randomly sample each action once (the exploration part). The partial information procedure MAB will pass to the full information procedure FIB the vector of losses received from its exploration steps. The full information procedure FIB will then return a new distribution over actions. The main part of the proof will be to relate the loss of the full information procedure FIB on the loss sequence it observes to the loss of the partial information procedure MAB on the real loss sequence.

We start by considering a full information procedure FIB that partitions the T time steps into K blocks, $B^1, \ldots, B^K$, where $B^i = \{(i-1)(T/K)+1, \ldots, i(T/K)\}$, and uses the same distribution in all the time steps of a block. (For simplicity we assume that K divides T.) Consider an R_K external regret minimization procedure FIB (over K time steps), which at the end of block i updates the distribution using the average loss vector, i.e., $c^\tau = \sum_{t \in B^\tau} \ell^t / |B^\tau|$. Let $C_i^K = \sum_{\tau=1}^K c_i^\tau$ and $C_{\min}^K = \min_i C_i^K$. Since FIB has external regret at most R_K, this implies that the loss of FIB, over the loss sequence c^τ, is at most $C_{\min}^K + R_K$. Since in every block B^τ the procedure FIB uses a single distribution p^τ, its loss on the entire loss sequence is:

$$L_{\text{FIB}}^T = \sum_{\tau=1}^K \sum_{t \in B^\tau} p^\tau \cdot \ell^t = \frac{T}{K} \sum_{\tau=1}^K p^\tau \cdot c^\tau \leq \frac{T}{K}\left[C_{\min}^K + R_K\right].$$

At this point it is worth noting that if $R_K = O(\sqrt{K \log N})$ the overall regret is $O((T/\sqrt{K})\sqrt{\log N})$, which is minimized at $K = T$, namely by having each block

[3] This reduction does not produce the best-known bounds for the partial information model (see, e.g., Auer et al., 2002 for better bounds) but is particularly simple and generic.

be a single time step. However, we will have an additional loss associated with each block (due to the sampling) which will cause the optimization to require that $K \ll T$.

The next step in developing the partial information procedure MAB is to use loss vectors that are not the "true average" but whose expectation is the same. More formally, the feedback to the full information procedure FIB will be a random variable vector $\hat{c}^\tau$ such that for any action i we have $E[\hat{c}_i^\tau] = c_i^\tau$. Similarly, let $\hat{C}_i^K = \sum_{\tau=1}^K \hat{c}_i^\tau$ and $\hat{C}_{\min}^K = \min_i \hat{C}_i^K$. (Intuitively, we will generate the vector $\hat{c}^\tau$ using sampling within a block.) This implies that for any block B^τ and any distribution p^τ we have

$$\frac{1}{|B^\tau|} \sum_{t \in B^\tau} p^\tau \cdot \ell^t \;=\; p^\tau \cdot c^\tau \;=\; \sum_{i=1}^N p_i^\tau c_i^\tau \;=\; \sum_{i=1}^N p_i^\tau E\big[\hat{c}_i^\tau\big] \tag{4.2}$$

That is, the loss of p^τ in B^τ is equal to its expected loss with respect to $\hat{c}^\tau$.

The full information procedure FIB observes the losses $\hat{c}^\tau$, for $\tau \in \{1, \ldots, K\}$. However, since $\hat{c}^\tau$ are random variables, the distribution p^τ is also a random variable that depends on the previous losses, i.e., $\hat{c}^1, \ldots, \hat{c}^{\tau-1}$. Still, with respect to any sequence of losses $\hat{c}^\tau$, we have that

$$\hat{C}_{\mathtt{FIB}}^K \;=\; \sum_{\tau=1}^K p^\tau \cdot \hat{c}^\tau \;\le\; \hat{C}_{\min}^K + R_K$$

Since $E[\hat{C}_i^K] = C_i^K$, this implies that

$$E\big[\hat{C}_{\mathtt{FIB}}^K\big] \le E\big[\hat{C}_{\min}^K\big] + R_K \le C_{\min}^K + R_K,$$

where we used the fact that $E[\min_i \hat{C}_i^K] \le \min_i E[\hat{C}_i^K]$ and the expectation is over the choices of $\hat{c}^\tau$.

Note that for any sequence of losses $\hat{c}^1, \ldots, \hat{c}^K$, both FIB and MAB will use the same sequence of distributions $p^1, \ldots, p^K$. From (4.2) we have that in any block B^τ the expected loss of FIB and the loss of MAB are the same, assuming they both use the same distribution p^τ. This implies that

$$E\big[C_{\mathtt{MAB}}^K\big] = E\big[\hat{C}_{\mathtt{FIB}}^K\big].$$

We now need to show how to derive random variables $\hat{c}^\tau$ with the desired property. This will be done by choosing randomly, for each action i and block B^τ, an exploration time $t_i \in B^\tau$. (These do not need to be independent over the different actions, so can easily be done without collisions.) At time t_i the procedure MAB will play action i (i.e., the probability vector with all probability mass on i). This implies that the feedback that it receives will be $\ell_i^{t_i}$, and we will then set $\hat{c}_i^\tau$ to be $\ell_i^{t_i}$. This guarantees that $E[\hat{c}_i^\tau] = c_i^\tau$.

So far we have ignored the loss in the exploration steps. Since the maximum loss is 1, and there are N exploration steps in each of the K blocks, the total loss in all the exploration steps is at most NK. Therefore we have

$$\begin{aligned} E\big[L_{\mathtt{MAB}}^T\big] &\le NK + (T/K)E\big[C_{\mathtt{MAB}}^K\big] \\ &\le NK + (T/K)\big[C_{\min}^K + R_K\big] \\ &= L_{\min}^T + NK + (T/K)R_K. \end{aligned}$$

By Theorem 4.6, there are external regret procedures that have regret $R_K = O(\sqrt{K \log N})$. By setting $K = (T/N)^{2/3}$, for $T \geq N$, we have the following theorem.

Theorem 4.17 *Given an $O(\sqrt{K \log N})$ external regret procedure* FIB *(for K time steps), there is a partial information procedure* MAB *that guarantees*

$$L^T_{\mathtt{MAB}} \leq L^T_{\min} + O(T^{2/3} N^{1/3} \log N),$$

where $T \geq N$.

4.7 On Convergence of Regret-Minimizing Strategies to Nash Equilibrium in Routing Games

As mentioned earlier, one natural setting for regret-minimizing algorithms is online routing. For example, a person could use such algorithms to select which of N available routes to use to drive to work each morning in such a way that his performance will be nearly as good as the best fixed route in hindsight, even if traffic changes arbitrarily from day to day. In fact, even though in a graph G, the number of paths N between two nodes may be exponential in the size of G, there are a number of external-regret minimizing algorithms whose running time and regret bounds are polynomial in the graph size. Moreover, a number of extensions have shown how these algorithms can be applied even to the partial-information setting where only the cost of the path traversed is revealed to the algorithm.

In this section we consider the game-theoretic properties of such algorithms in the Wardrop model of traffic flow. In this model, we have a directed network $G = (V, E)$, and one unit flow of traffic (a large population of infinitesimal users that we view as having one unit of volume) wanting to travel between two distinguished nodes v_{start} and v_{end}. (For simplicity, we are considering just the single-commodity version of the model.) We assume each edge e has a cost given by a *latency function* ℓ_e that is some nondecreasing function of the amount of traffic flowing on edge e. In other words, the time to traverse each edge e is a function of the amount of congestion on that edge. In particular, given some flow f, where we use f_e to denote the amount of flow on a given edge e, the cost of some path P is $\sum_{e \in P} \ell_e(f_e)$ and the average travel time of all users in the population can be written as $\sum_{e \in E} \ell_e(f_e) f_e$. A flow f is at Nash equilibrium if all flow-carrying paths P from v_{start} to v_{end} are minimum-latency paths *given* the flow f.

Chapter 18 considers this model in much more detail, analyzing the relationship between latencies in Nash equilibrium flows and those in globally optimum flows (flows that minimize the total travel time averaged over all users). In this section we describe results showing that if the users in such a setting are adapting their paths from day to day using external-regret minimizing algorithms (or even if they just happen to experience low-regret, regardless of the specific algorithms used) then flow will approach Nash equilibrium. Note that a Nash equilibrium is precisely a set of static strategies that are all no-regret with respect to each other, so such a result seems natural; however, there are many simple games for which regret-minimizing algorithms do *not* approach Nash equilibrium and can even perform much worse than any Nash equilibrium.

Specifically, one can show that if each user has regret $o(T)$, or even if just the average regret (averaged over the users) is $o(T)$, then flow approaches Nash equilibrium in the sense that a $1 - \epsilon$ fraction of days t have the property that a $1 - \epsilon$ fraction of the users that day experience travel time at most ϵ larger than the best path for that day, where ϵ approaches 0 at a rate that depends polynomially on the size of the graph, the regret-bounds of the algorithms, and the maximum slope of any latency function. Note that this is a somewhat nonstandard notion of convergence to equilibrium: usually for an "ϵ-approximate equilibrium" one requires that *all* participants have at most ϵ incentive to deviate. However, since low-regret algorithms are allowed to occasionally take long paths, and in fact algorithms in the MAB model *must* occasionally explore paths they have not tried in a long time (to avoid regret if the paths have become much better in the meantime), the multiple levels of hedging are actually *necessary* for a result of this kind.

In this section we present just a special case of this result. Let $\mathcal{P}$ denote the set of all simple paths from v_{start} to v_{end} and let f^t denote the flow on day t. Let $C(f) = \sum_{e \in E} \ell_e(f_e) f_e$ denote the cost of a flow f. Note that $C(f)$ is a weighted average of costs of paths in $\mathcal{P}$ and in fact is equal to the average cost of all users in the flow f. Define a flow f to be *ϵ-Nash* if $C(f) \leq \epsilon + \min_{P \in \mathcal{P}} \sum_{e \in P} \ell_e(f_e)$; that is, the average incentive to deviate over all users is at most ϵ. Let $R(T)$ denote the average regret (averaged over users) up through day T, so

$$R(T) \equiv \sum_{t=1}^{T} \sum_{e \in E} \ell_e\big(f_e^t\big) f_e^t - \min_{P \in \mathcal{P}} \sum_{t=1}^{T} \sum_{e \in P} \ell_e\big(f_e^t\big).$$

Finally, let T_ϵ denote the number of time steps T needed so that $R(T) \leq \epsilon T$ for all $T \geq T_\epsilon$. For example the RWM and PW algorithms discussed in Section 4.3 achieve $T_\epsilon = O(\frac{1}{\epsilon^2} \log N)$ if we set $\eta = \epsilon/2$. Then we will show the following.

Theorem 4.18 *Suppose the latency functions ℓ_e are linear. Then for $T \geq T_\epsilon$, the average flow $\hat{f} = \frac{1}{T}(f^1 + \cdots + f^T)$ is ϵ-Nash.*

PROOF From the linearity of the latency functions, we have for all e, $\ell_e(\hat{f}_e) = \frac{1}{T} \sum_{t=1}^{T} \ell_e(f_e^t)$. Since $\ell_e(f_e^t) f_e^t$ is a convex function of the flow, this implies

$$\ell_e(\hat{f}_e)\hat{f}_e \leq \frac{1}{T} \sum_{t=1}^{T} \ell_e\big(f_e^t\big) f_e^t.$$

Summing over all e, we have

$$\begin{aligned} C(\hat{f}) &\leq \frac{1}{T} \sum_{t=1}^{T} C(f^t) \\ &\leq \epsilon + \min_P \frac{1}{T} \sum_{t=1}^{T} \sum_{e \in P} \ell_e\big(f_e^t\big) \quad \text{(by definition of } T_\epsilon\text{)} \\ &= \epsilon + \min_P \sum_{e \in P} \ell_e(\hat{f}_e). \qquad \text{(by linearity)} \end{aligned}$$

□

This result shows the time-average flow is an approximate Nash equilibrium. This can then be used to prove that *most* of the f^t must in fact be approximate Nash. The key idea here is that if the cost of any edge were to fluctuate wildly over time, then that would imply that most of the users of that edge experienced latency substantially greater than the edge's average cost (because more users are using the edge when it is congested than when it is not congested), which in turn implies they experience substantial regret. These arguments can then be carried over to the case of general (nonlinear) latency functions.

4.7.1 Current Research Directions

In this section we sketch some current research directions with respect to regret minimization.

Refined regret bounds: The regret bounds that we presented depend on the number of time steps T, and are independent of the performance of the best action. Such bounds are also called *zero-order* bounds. More refined *first-order* bounds depend on the loss of the best action, and *second-order* bounds depend on the sum of squares of the losses (such as Q_k^T in Theorem 4.6). An interesting open problem is to get an external regret that is proportional to the empirical variance of the best action. Another challenge is to reduce the prior information needed by the regret minimization algorithm. Ideally, it should be able to learn and adapt to parameters such as the maximum and minimum loss. See Cesa-Bianchi et al. (2005) for a detailed discussion of those issues.

Large actions spaces: In this chapter we assumed the number of actions N is small enough to be able to list them all, and our algorithms work in time proportional to N. However, in many settings N is exponential in the natural parameters of the problem. For example, the N actions might be all simple paths between two nodes s and t in an n-node graph, or all binary search trees on $\{1, \ldots, n\}$. Since the full information external regret bounds are only logarithmic in N, from the point of view of information, we can derive polynomial regret bounds. The challenge is whether in such settings we can produce computationally efficient algorithms.

There have recently been several results able to handle broad classes of problems of this type. Kalai and Vempala (2003) give an efficient algorithm for any problem in which (a) the set X of actions can be viewed as a subset of R^n, (b) the loss vectors ℓ are linear functions over R^n (so the loss of action x is $\ell \cdot x$), and (c) we can efficiently solve the *offline* optimization problem $\text{argmin}_{x \in S}[x \cdot \ell]$ for any *given* loss vector ℓ. For instance, this setting can model the path and search-tree examples above.[4] Zinkevich (2003) extends this to *convex* loss functions with a projection oracle, and there is substantial interest in trying to broaden the class of settings that efficient regret-minimization algorithms can be applied to.

[4] The case of search trees has the additional issue that there is a rotation cost associated with using a different action (tree) at time $t + 1$ than that used at time t. This is addressed in Kalai and Vempala (2003) as well.

Dynamics: It is also very interesting to analyze the *dynamics* of regret minimization algorithms. The classical example is that of swap regret: when all the players play swap regret-minimization algorithms, the empirical distribution converges to the set of correlated equilibria (Section 4.4). We also saw convergence in two-player zero-sum games to the minimax value of the game (Section 4.4), and convergence to Nash equilibrium in a Wardrop-model routing game (Section 4.7). Further results on convergence to equilibria in other settings would be of substantial interest. At a high level, understanding the dynamics of regret-minimization algorithms would allow us to better understand the strengths and weaknesses of using such procedures. For more information on learning in games, see the book by Fudenberg and Levine (1998).

4.8 Notes

Hannan (1957) was the first to develop algorithms with external regret sublinear in T. Later, motivated by machine learning settings in which N can be quite large, algorithms that furthermore have only a logarithmic dependence on N were developed by Littlestone and Warmuth (1994), and extended by a number of researchers (Cesa-Bianchi et al., 1997; Freund and Schapire, 1997, 1999). In particular, the Randomized Weighted Majority algorithm and Theorem 4.5 are from Littlestone and Warmuth (1994) and the Polynomial Weights algorithm and Theorem 4.6 is from Cesa-Bianchi et al. (2005). Computationally efficient algorithms for generic frameworks that model many settings in which N may be exponential in the natural problem description (such as considering all s-t paths in a graph or all binary search trees on n elements) were developed in Kalai and Vempala (2000) and Zinkevich (2003).

The notion of internal regret and its connection to correlated equilibrium appear in Foster and Vohra (1998) and Hart and Mas-Colell (2000) and more general modification rules were considered in Lehrer (2003). A number of specific low internal regret algorithms were developed by a number of researcher (Blum and Mansour, 2005; Cesa-Bianchi and Lugosi, 2003; Foster and Vohra, 1997, 1998, 1999; Hart and Mas-Colell, 2003; Stoltz and Lugosi, 2005). The reduction in Section 4.5 from external to swap regret is from Blum and Mansour (2005).

Algorithms with strong external regret bounds for the partial information model are given in Auer et al. (2002) , and algorithms with low internal regret appear in Blum and Mansour (2005) and Cesa-Bianchi et al. (2006). The reduction from full information to partial information in Section 4.6 is in the spirit of algorithms of Awerbuch and Mansour (2003) and Awerbuch and Kleinberg (2004). Extensions of the algorithm of Kalai and Vempala (2003) to the partial information setting appear in Awerbuch and Kleinberg (2004), Dani and Hayes (2006) and McMahan and Blum (2004). The results in Section 4.7 on approaching Nash equilibria in routing games are from Blum et al. (2006).

Bibliography

P. Auer, N. Cesa-Bianchi, Y. Freund, and R.E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comp.*, 32(1):48–77, 2002.

B. Awerbuch and R.D. Kleinberg. Adaptive routing with end-to-end feedback: Distributed learning and geometric approaches. In *Symp. on Theory of Computing*, pp. 45–53, 2004.

B. Awerbuch and Y. Mansour. Adapting to a reliable network path. In *PODC*, pp. 360–367, 2003.

A. Blum, E. Even-Dar, and K. Ligett. Routing without regret: On convergence to nash equilibria of regret-minimizing algorithms in routing games. In *Princ. Distributed Comp.*, 2006.

A. Blum and Y. Mansour. From external to internal regret. In *Conf. on Learning Theory*, 2005.

A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

N. Cesa-Bianchi, Y. Freund, D.P. Helmbold, D. Haussler, R.E. Schapire, and M.K. Warmuth. How to use expert advice. *J. ACM*, 44(3):427–485, 1997.

N. Cesa-Bianchi and G. Lugosi. Potential-based algorithms in on-line prediction and game theory. *Mach. Learn.*, 51(3):239–261, 2003.

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning and Games*. Cambridge University Press, 2006.

N. Cesa-Bianchi, G. Lugosi, and G. Stoltz. Regret minimization under partial monitoring. *Math. of O.R.* (to appear), 2006.

N. Cesa-Bianchi, Y. Mansour, and G. Stoltz. Improved second-order bounds for prediction with expert advice. In *Conf. on Learning Theory*, 2005.

V. Dani and T.P. Hayes. Robbing the bandit: Less regret in online geometric optimization against an adaptive adversary. In *Symp. on Descrete Algorithms*, pp. 937–943, 2006.

D. Foster and R. Vohra. Calibrated learning and correlated equilibrium. *Games Econ. Behav.*, 21:40–55, 1997.

D. Foster and R. Vohra. Asymptotic calibration. *Biometrika*, 85:379–390, 1998.

D. Foster and R. Vohra. Regret in the on-line decision problem. *Games Econ. Behav.*, 29:7–36, 1999.

Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comp. System Sci.*, 55(1):119–139, 1997.

Y. Freund and R.E. Schapire. Adaptive game playing using multiplicative weights. *Games Econ. Behav.*, 29:79–103, 1999.

D. Fudenberg and D.K. Levine. *The Theory of Learning in Games*. MIT Press, 1998.

J. Hannan. Approximation to bayes risk in repeated plays. In M. Dresher, A. Tucker, and P. Wolfe, editors, *Contributions to the Theory of Games*, 3:97–139, Princeton University Press, 1957.

S. Hart and A. Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68:1127–1150, 2000.

A. Kalai and S. Vempala. Efficient algorithms for online decision problems. In *Conf. on Learning Theory*, pp. 26–40, 2003.

E. Lehrer. A wide range no-regret theorem. *Games Econ. Behav.*, 42:101–115, 2003.

N. Littlestone and M.K. Warmuth. The weighted majority algorithm. *Informat. Comput.*, 108:212–261, 1994.

H.B. McMahan and A. Blum. Online geometric optimization in the bandit setting against an adaptive adversary. In *Proc. 17th Annual Conference on Learning Theory*, pp. 109–123, 2004.

G. Stoltz and G. Lugosi. Internal regret in on-line portfolio selection. *Mach. Learn. J.*, 59:125–159, 2005.

G. Owen. *Game Theory*. Academic Press, 1982.

D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Comm. ACM*, 28:202–208, 1985.

M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proc. Intl. Conf. Machine Learning*, 928–936, 2003.

Exercises

4.1 Show that swap regret is at most N times larger than internal regret.

4.2 Show an example (even with $N = 3$) where the ratio between the external and swap regret is unbounded.

4.3 Show that the RWM algorithm with update rule $w_i^t = w_i^{t-1}(1-\eta)^{\ell_i^{t-1}}$ achieves the same external regret bound as given in Theorem 4.6 for the PW algorithm, for losses in $[0, 1]$.

4.4 Consider a setting where the payoffs are in the range $[-1, +1]$, and the goal of the algorithm is to maximize its payoff. Derive a modified PW algorithm whose external regret is $O(\sqrt{Q_{\max}^T \log N} + \log N)$, where $Q_{\max}^T \geq Q_k^T$ for $k \in X_i$.

4.5 Show a $\Omega(\sqrt{T \log N})$ lower bound on external regret, for the case that $T \geq N$.

4.6 Improve the swap regret bound to $O(\sqrt{NT \log N})$. Hint: Use the observation that the sum of the losses of all the A_i is bounded by T.

4.7 **(Open Problem)** Does there exist an $\Omega(\sqrt{TN \log N})$ lower bound for swap regret?

4.8 Show that if a player plays algorithm RWM (or PW) then it gives ϵ-dominated actions small weight. Also, show that there are cases in which the external regret of a player can be small, yet it gives ϵ-dominated actions high weight.