Exploring fair machine learning in sequential prediction and supervised learning

by

Sajjad Azami
B.Sc., Amirkabir University of Technology, 2018

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

Exploring fair machine learning in sequential prediction and supervised learning

by

Sajjad Azami
B.Sc., Amirkabir University of Technology, 2018

Supervisory Committee

Dr. Nishant Mehta, Supervisor
(Department of Computer Science, University of Victoria)

Dr. Cristóbal Guzmán, Member of Supervisory Committee
(Department of Computer Science, University of Victoria)

Supervisory Committee

Dr. Nishant Mehta, Supervisor
(Department of Computer Science, University of Victoria)

Dr. Cristóbal Guzmán, Member of Supervisory Committee
(Department of Computer Science, University of Victoria)

ABSTRACT

Algorithms that are being used in sensitive contexts such as deciding to give a job offer or giving inmates parole should be accurate as well as being non-discriminatory. The latter is important especially due to emerging concerns about automatic decision making being unfair to individuals belonging to certain groups. The machine learning literature has seen a rapid evolution in research on this topic. In this thesis, we study various problems in sequential decision making motivated by challenges in algorithmic fairness. As part of this thesis, we modify the fundamental framework of prediction with expert advice. We assume a learning agent is making decisions using the advice provided by a set of experts while this set can shrink. In other words, experts can become unavailable due to scenarios such as emerging anti-discriminatory laws prohibiting the learner from using experts detected to be unfair. We provide efficient algorithms for this setup, as well as a detailed analysis of the optimality of them. Later we explore a problem concerned with providing any-time fairness guarantees using the well-known exponential weights algorithm, which leads to an open question about a lower bound on the cumulative loss of exponential weights algorithm. Finally, we introduce a novel fairness notion for supervised learning tasks motivated by the concept of envy-freeness. We show how this notion might bypass certain issues of existing fairness notions such as equalized odds. We provide solutions for a simplified version of this problem and insights to deal with further challenges that arise by adopting this notion.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

First, I would like to thank the best teacher I have had, my advisor Nishant Mehta. His exceptional patience in teaching me how to do research helped me gain this invaluable experience. I want to thank Cristóbal Guzmán for his substantial guidance; this thesis would not be possible without him. I am also grateful to have Bo Waggoner as my examiner and having the chance to hear his interesting comments and ideas.

Next, thanks to my colleague and labmate, Hamid Shayestehmanesh, for his vital contribution to Chapter 3 of this work, in addition to allowing me to steal his tea all the time! Also, thanks to my other labmates, Bingshan and Sharoff, for useful discussions we had over the past two years.

I would like to thank the staff at the computer science department, namely Wendy Beggs, Kath Milinazzo and Nancy Chan, for their help with every administrative and financial matter during my time here.

Finally, I am grateful for my family and all the incredible friends I have for always being there for me.

Chapter 1

Introduction

We often deal with problems and tasks that can be modeled as a sequence of decisions. To perform well, one naturally would like to learn from each decision moving forward. As an example, consider we are investing in the stock market. We would like to maintain knowledge about the market and hopefully improve this knowledge by learning from past decisions. This sequential setting is particularly in contrast with a situation where we have some data to learn from and make predictions on a new, randomly drawn set of examples; this is often referred to as the batch setting. In sequential prediction, examples can be chosen adversarially, making it rather difficult to minimize the loss or mistakes as it is usually done in the batch setting. Instead, it makes more sense to try to make decisions such that we will not regret them later. We will formally define the notion of "regret" later in this thesis (see Chapter 2).

In machine learning, this sequential decision making is modeled in a setting referred to as online learning. Most of the contributions in this thesis are related to problems framed in this sequential setting. To explain our framework more precisely, imagine we are deciding whether to admit a student into a program or not. We, as the learner (hereafter referred to as Learner), are making a final decision about each applicant as we go through applications. Now to be more realistic, imagine there is a board that gives us advice about each applicant. For simplicity, assume for each applicant, each member of the board (hereafter referred to as an "expert") gives us a simple "yes" or "no" response to the question of whether this applicant should be admitted or not. This way, the task of learning is reduced to learn about these experts based on their performance. In other words, Learner needs to learn how good each expert is and trust their response proportional to that.

This framework is referred to as *prediction with expert advice*, which is central to

most parts of this study. Naturally, in any learning task, Learner wants to predict as accurately as possible. However, our main concern is not just to obtain accurate algorithms, but also fair ones. To explain the notion of "fair algorithms", let us use another example. Imagine we are deciding whether an incarcerated individual should be given parole or not. As we mentioned earlier, Learner is using expert advice based on experts' *performance*. In the parole example, a good performance is giving parole to individuals who will not commit crime after returning to society and not giving parole to those who will end up committing crime again. On the other hand, we say Learner made a mistake if they decide to give parole to an individual that will commit a crime later (*false positive* error) or refuse to give parole to an inmate that in the hindsight, if given parole, was not going to commit crime (*false negative* error). These decisions, however, might be "unfair" to certain groups or individuals. For example, imagine examples that belong to certain groups (have a common attribute) have higher false negative rates than other groups. This is a case where the algorithm is discriminating against this group. The same argument applies to similar scenarios, such as the already mentioned college admission problem and job application where certain groups might be discriminated against. The notion that measures how fair these decisions are can vary and the results will have different implications. Our motivation for this thesis's results is to provide algorithms for certain scenarios with some types of fairness guarantee.

In Chapter 3[1], we look into a case of prediction with expert advice framework that is motivated by scenarios where Learner is not allowed to use experts that are found to be discriminating. This setup is tied closely to a well-studied problem referred to as *sleeping experts*. In sleeping experts, the set of experts that Learner uses for prediction can change during the game. However, we study a modified version of sleeping experts, where we allow this set only to shrink. The sleeping experts setting is known to be challenging, particularly in terms of computational efficiency; however, we provide efficient algorithms with optimal regret guarantees to our modified problem.

In Chapter 4, we attempt to provide fairness guarantees for a fundamental algorithm in prediction with expert advice. Our setup is close to the one in Blum et al. (2018), where they guarantee fairness (with respect to a notion to be defined later) when all the experts are fair in isolation. However, they guarantee fairness only at the very end of the game, which raises the question of whether this analysis can be extended

[1]The results in this chapter is published at NeurIPS 2019 (Shayestehmanesh et al. (2019)). Algorithms developed for dying experts are due to my co-author, Hamid.

to guarantee fairness in *every* round of the decision making process. This results in a very interesting question about a fundamental algorithm in prediction with expert advice (Hedge algorithm), which, as far as we are aware, has not been answered yet.

In a somewhat different setup, in Chapter 5, we depart from online learning to supervised batch setting, where examples are drawn according to an unknown distribution. We adopt a new notion of fairness, which is motivated by the following argument: imagine examples belong to two groups and for each group, we have a hypothesis that best predicts examples from that group, but Learner is expected to output a single hypothesis to be used for both groups. Our notion of fairness ensures Learner chooses a hypothesis that no group envies the other. This envy-based notion results in a min-max objective, and we provide solution towards solving the problem as well as insights for interesting challenges and questions arising from the setup.

We will introduce each part formally in their respective chapters, but before proceeding further, we give some background for fundamental topics needed for the rest of the manuscript in Chapter 2.

Chapter 2

Background

2.1 Prediction with Expert Advice

In prediction with expert advice (Littlestone and Warmuth (1994); Vovk (1990, 1998)), Learner has access to K experts, and at each round t, a sample is given by the environment (hereafter referred to as Nature) and each expert outputs a prediction f_t. Learner then predicts an outcome $\hat{y}_t \in \mathcal{D}$ where $\mathcal{D}$ is the decision space. Finally, Nature reveals the label $y_t \in \mathcal{Y}$ where $\mathcal{Y}$ is the outcome space and Learner suffers loss as $\ell(\hat{y}_t, y_t)$ where $\ell(y, y') : \mathcal{D} \times \mathcal{Y} \to \mathbb{R}$. This game repeats for T rounds. The protocol is summarized in Figure 2.1 below.

For round $t = 1, 2, \ldots T$

1. Learner receives expert advice $f_{1,t}, f_{2,t}, \ldots, f_{K,t}$
2. Learner predicts $\hat{y}_t$
3. Nature selects outcome y_t
4. Learner suffers $\hat{\ell}_t = \ell(\hat{y}_t, y_t)$ and each expert suffers loss $\ell_{j,t} := \ell(f_{K,t}, y_t)$ for $j \in [K]$

Figure 2.1: Prediction with expert advice (PEA) protocol

The performance of Learner over T rounds is scored with respect to the classical

notion of regret defined as

$$R = \sum_{t=1}^{T} \hat{\ell}_t - \min_{j \in [K]} \sum_{t=1}^{T} \ell_{j,t}. \tag{2.1}$$

Intuitively, $\arg\min_{j \in [K]} \sum_{t=1}^{T} \ell_{j,t}$ is the *best expert in the hindsight*. That is, if Learner was to predict according to only one expert, in hindsight this expert would be the best choice. Therefore, this notion of regret represents the regret to the best.

Learner's prediction $\hat{y}_t$ can be a probability distribution p_t on the experts. This way, Learner's loss $\hat{\ell}_t$ becomes $\mathsf{E}_{j \sim p_t}[\ell_{j,t}]$. Now the question is how to maintain/update the probability distribution at each round. Prior to answer this question, one needs to find out what rate of growth for regret is satisfying. From here onward, we assume the losses are in the interval $[0, 1]$. Also, note that we have not made any assumptions about Nature. Often in the literature, Nature's power in selecting the losses for experts is decided by making two distinctions:

- *Stochastic* vs *Adversarial*: In the stochastic setting, the losses are assumed to be drawn i.i.d. from a distribution, whereas in the adversarial setting, no such assumption is made.

- *Oblivious* vs *Adaptive*: An oblivious adversary fixes a loss sequence $\boldsymbol{\ell}_j := (\ell_{j,1}, \ldots, \ell_{j,T})$ for $j \in [K]$ beforehand, while in the adaptive case, Nature can react to the decisions of Learner. In other words, at round t, $\ell_{j,t}$ can depend on $p_1, \ldots, p_{t-1}$ for any $j \in [K]$.

With these explanations, for the rest of this thesis, unless stated otherwise, we assume Nature is an adversarial adaptive one, which is the most general (and therefore hardest as viewed by Learner) case.

Given this, one can easily verify that achieving average regret linear in the length of time horizon ($\mathcal{O}(T)$) is trivial. Therefore, we are interested in a rate sublinear in T; meaning, as $T \to \infty$, average regret approaches zero. An algorithm that obtains this rate is referred to as a *no-regret* algorithm. Now, back to the question of what Learner should do to obtain such a rate. One fundamental algorithm called *Hedge*, also known as exponential weights, does so. In the next section, we introduce this algorithm and its analysis as it is central for the remainder of this thesis.

2.2 Exponential Weights (Hedge)

The exponential weights algorithm is due to Freund and Schapire (1997). In exponential weights, Learner maintains weights $w_{j,t}$ for $j \in [K]$ and $t = 1, \ldots, T$. As the name of the algorithm suggests, these weights are updated exponential in cumulative loss of each expert. More specifically, $w_{j,t}$ decays by $e^{-\eta \ell_{j,t}}$ for the next round. We formally introduce the algorithm in Algorithm 1.

Algorithm 1: Exponential Weights (Hedge)

Choose $\eta > 0$
Set $w_{j,0} = 1$ for $j \in [K]$
for *t = 1, 2, . . . , T* **do**
  Set $p_{j,t} = \frac{w_{j,t-1}}{\sum_{j \in [K]} w_{j,t-1}}$ for $j \in [K]$
  Receive loss vector $\boldsymbol{\ell}_t = (\ell_{1,t}, \ldots, \ell_{j,t})$
  Suffer loss $\boldsymbol{\ell}_t \cdot \boldsymbol{p}_t$
  Update weights as $w_{j,t} = w_{j,t-1} e^{-\eta \ell_{j,t}}$ for $j \in [K]$

Analysis. In the following, we show an analysis of the Hedge algorithm's regret. For a deeper understanding, we refer the reader to Cesa-Bianchi and Lugosi (2006)[Chapter 2].

Theorem 1 *For a loss function ℓ that is convex in its first argument and bounded in $[0, 1]$, for any sequence of loss Nature reveals, Hedge with learning rate $\eta > 0$ guarantees:*

$$\sum_{t=1}^{T} \hat{\ell}_t \leq \min_{j \in [K]} \sum_{t=1}^{T} \ell_{j,t} + \frac{\ln K}{\eta} + \frac{T\eta}{8}$$

Before showing the proof, note that setting $\eta = \sqrt{\frac{8 \ln K}{T}}$ optimizes the upper bound and results in $R \leq \sqrt{\frac{T \ln K}{2}}$.

Proof For convenience, let us denote by W_t the sum of weights over all the experts.

$$W_t = \sum_{j \in [K]} w_{j,t}$$

Denote the cumulative loss of an expert until round t by $L_{j,t} := \sum_{s=1}^{t} \ell_{j,s}$. It is easy to verify that:

$$W_t = \sum_{j \in [K]} e^{-\eta L_{j,t}}$$

The strategy is to obtain a lower and upper bound on the term $\ln \frac{W_T}{W_0}$.

Lower bound. To obtain the lower bound, we use the fact that sum of non-negative terms is lower bounded by their maximum term:

$$\begin{aligned}
\ln \frac{W_T}{W_0} &= \ln \left(\sum_{j \in [K]} e^{-\eta L_{j,T}} \right) - \ln K \\
&\geq \ln \left(\max_{j \in [K]} e^{-\eta L_{j,T}} \right) - \ln K \\
&= -\eta \underbrace{\min_{j \in [K]} L_{j,T}}_{\text{loss of the best expert}} - \ln K
\end{aligned} \tag{2.2}$$

Upper bound. To achieve the upper bound, note that:

$$\ln \frac{W_T}{W_0} = \sum_{t=1}^{T} \ln \frac{W_t}{W_{t-1}}. \tag{2.3}$$

Therefore, it suffices to upper bound each term $\ln \frac{W_t}{W_{t-1}}$. For this, we use the well known Hoeffding's Lemma (re-stated below as Lemma 1) due to Hoeffding (1994). Using Lemma 1 and unpacking the definition of W_t, we get:

$$\begin{aligned}
\ln \frac{W_t}{W_{t-1}} &= \ln \frac{\sum_{j \in [k]} e^{-\eta L_{j,t}}}{\sum_{j \in [k]} w_{j,t-1}} \\
&= \ln \frac{\sum_{j \in [k]} e^{-\eta L_{j,t-1}} e^{-\eta \ell_{j,t}}}{\sum_{j \in [k]} w_{j,t-1}} \\
&= \ln \frac{\sum_{j \in [k]} w_{j,t-1} e^{-\eta \ell_{j,t}}}{\sum_{j \in [k]} w_{j,t-1}} \\
&= \ln \mathsf{E}_{j \sim p_t} [e^{-\eta \ell_{j,t}}] \\
&\leq -\eta \, \mathsf{E}_{j \sim p_t} [\ell_{j,t}] + \frac{\eta^2}{8}.
\end{aligned} \tag{2.4}$$

Plugging (2.4) into (2.3), we get:

$$\ln \frac{W_T}{W_0} \leq -\eta \underbrace{\sum_{t=1}^{T} \mathsf{E}_{j \sim p_t} [\ell_{j,t}]}_{\text{loss of Hedge}} + \frac{T\eta^2}{8} \tag{2.5}$$

Finally, combining the lower and upper bound in (2.2) and (2.5), we get:

$$-\eta \min_{j\in[K]} L_{j,T} - \ln K \leq -\eta \sum_{t=1}^{T} \mathsf{E}_{j\sim p_t}[\ell_{j,t}] + \frac{T\eta^2}{8}$$

and the result follows by rearranging the above. ∎

Lemma 1 (Hoeffding (1994)) *Let X be a random variable satisfying $\mathsf{E}[X] = 0$ and $a \leq X \leq b$. For any $\lambda \in \mathbb{R}$, we have:*

$$\log \mathsf{E}[e^{\lambda X}] \leq \frac{\lambda^2 (b-a)^2}{8}.$$

2.3 Time-varying Learning Rate

As one might already have noticed, the guarantee in Theorem 1 needs the learning rate η to be set beforehand to $\sqrt{\frac{8\ln K}{T}}$. This means Learner should a priori know the number of rounds T, while this might be an unreasonable assumption in real-world scenarios. In other words, the bound in Theorem 1 does not hold uniformly over the time horizon. Dealing with this issue is particularly important for the problem in Chapter 4.

To tackle this problem, a simple solution is to use what is called "doubling trick". The idea is to make a guess about the time horizon and set the learning rate according to that, and as soon as t gets equal to our guess, we double the guess. As an example, we start assuming $T = 1$, and set learning rate $\eta = \sqrt{\frac{8\ln K}{1}}$. Then in round 2 (if the game gets there), we double the guess to 2 and learning rate becomes $\eta = \sqrt{\frac{8\ln K}{2}}$, then for round 3 and 4 it becomes $\eta = \sqrt{\frac{8\ln K}{4}}$ and so forth. This divides the time horizon to r epochs of size $2^0, 2^1, 2^2, \ldots$. For this strategy, one can obtain the bound below:

$$\sum_{t=1}^{T} \hat{\ell}_t \leq \min_{j\in[K]} \sum_{t=1}^{T} \ell_{j,t} + \frac{\sqrt{2}}{\sqrt{2}-1}\sqrt{\frac{T\ln K}{2}} \tag{2.6}$$

which is worse than what we had before by constant factor. A more interesting solution would be to use a time-varying learning rate. That is, we choose $\eta = \sqrt{\frac{8\ln K}{t}}$. Using the analysis in Cesa-Bianchi and Lugosi (2006)[Theorem 2.3], we get the following guarantee.

Theorem 2 (Theorem 2.3 Cesa-Bianchi and Lugosi (2006)) *For a loss function ℓ that is convex in its first argument and bounded in $[0,1]$, for any sequence of loss Nature reveals, Hedge with learning rate $\eta_t := \sqrt{\frac{8 \ln K}{t}}$ satisfies:*

$$\sum_{t=1}^{T} \hat{\ell}_t \leq \min_{j \in [K]} \sum_{t=1}^{T} \ell_{j,t} + 2\sqrt{\frac{T}{2} \ln K} + \sqrt{\frac{\ln K}{8}}$$

Now that we have reviewed the fundamentals needed, we are ready to introduce a modified version of prediction with expert advice in the next Chapter. As mentioned earlier, the setting in Chapter 3 is motivated by the following argument about fairness: If the experts that are being used for prediction are found to be unfair, Learner might have to avoid using their advice. This gives rise to our modification of prediction with expert advice, what we refer to as *dying experts*.

Chapter 3

Dying Experts

3.1 Introduction

Decision-theoretic online learning (DTOL) (Littlestone and Warmuth (1994); Vovk (1990, 1998); Freund and Schapire (1997)) is a sequential game between a learning agent (*Learner*) and Nature. In each round, Learner plays a probability distribution over a fixed set of experts and suffers loss accordingly. However, in wide range of applications, this "fixed" set of actions shrinks as the game goes on. One way this can happen is because experts either get disqualified or expire over time; a key scenario of contemporary relevance is in contexts where experts that discriminate are prohibited from being used due to existing (or emerging) anti-discrimination laws. Two prime examples are college admissions and deciding whether incarcerated individuals should be granted parole; here the agent may rely on predictions from a set of experts in order to make decisions, and naturally experts detected to be discriminating against certain groups should not be played anymore. However, the standard DTOL setting does not directly adapt to this case, i.e., for a given round it does not make sense nor may it even be possible to compare Learner's performance to an expert or action that is no longer available.

Motivated by cases where the set of experts can change, a reasonable benchmark is the *ranking regret* (Kleinberg et al. (2010); Kale et al. (2016)), for which Learner competes with the best ordering of the actions (see (3.1) in Section 3.2 for a formal definition). The situation where the set of available experts can change in each round is known as the *sleeping experts* setting, and unfortunately, it appears to be computationally hard to obtain a no-regret algorithm in the case of adversarial payoffs

(losses in our setting) and adversarial availability of experts (Kanade and Steinke (2014)). This motivates the question of whether the optimal regret bounds can be achieved efficiently for the case where the set of experts can only shrink, which we will refer to as the "dying experts" setting. Applying the results of Kleinberg et al. (2010) to the dying experts problem only gives $\mathcal{O}(\sqrt{TK \log K})$ regret, for K experts and T rounds, and their strategy is computationally inefficient.

In more detail, the strategy in Kleinberg et al. (2010) is to define a permutation expert (our terminology) that is identified by an ordering of experts, where a permutation expert's strategy is to play the first awake expert in the ordering. They then run Hedge (Freund and Schapire (1997)) on the set of all possible permutation experts over K experts. Although this strategy competes with the best ordering, the per-round computation of running Hedge on $K!$ experts is $\mathcal{O}(K^K)$ if naïvely implemented, and the results of Kanade and Steinke (2014) suggest that no efficient algorithm — one that uses computation $\text{poly}(K)$ per round — can obtain regret that simultaneously is $o(T)$ and $\text{poly}(K)$.

However, in the dying experts setting, we show that many of these $K!$ orderings are redundant and only $\mathcal{O}(2^K)$ of them are "effective". The notion of effective experts (formally defined in Section 3.3) is used to refer to a minimal set of orderings such that each ordering in the set will behave uniquely in hindsight. The behavior of an ordering is defined as how it uses the initial experts in its predictions over T rounds. Interestingly, it turns out that this structure also allows for an efficient implementation of Hedge which, as we show, obtains optimal regret in the dying experts setting. The key idea that enables an efficient implementation is as follows. Our algorithms group orderings with identical behavior into one group, where there can be at most K groups at each round. When an expert dies, the orderings in one of the groups are forced to predict differently and therefore have to redistribute to the other groups. This splitting and rejoining behavior occurs in a fixed pattern which enables us to efficiently keep track of the weight associated with each group.

In certain scenarios, Learner might be aware of the order in which the experts will become unavailable. For example, in online advertising, an ad broker has contracts with their providers and these contracts may expire in an order known to Learner. Therefore, we will study the problem in two different settings: when Learner is aware of this order and when it is not.

Contributions of Chapter 3 Our first main result is an upper bound on the number of effective experts (Theorem 3); this result will be used for our regret upper bound in the known order case. Also, in preparation for our lower bound results, we prove a fully non-asymptotic lower bound on the minimax regret for DTOL (Theorem 4). Our main lower bounds contributions are minimax lower bounds for both the unknown and known order of dying cases (Theorems 5 and 7). In addition, we provide strategies to achieve optimal upper bounds for unknown and known order of dying (Theorems 6 and 8 respectively), along with efficient algorithms for each case. This is in particular interesting since, in the framework of sleeping experts, the results of Kanade and Steinke (2014) suggest that no-regret learning is computationally hard, but we show that it is efficiently achievable in the restricted problem. Finally, in Section 3.5.3, we show how to generalize our algorithms to other algorithms with adaptive learning rates, either adapting to unknown T or achieving far greater forms of adaptivity like in AdaHedge and FlipFlop (de Rooij et al. (2014)).

3.2 Background and related work

The DTOL setting (Freund and Schapire (1997)) is a variant of prediction with expert advice (Littlestone and Warmuth (1994); Vovk (1990, 1998)) in which Learner receives an example x_t in round t and plays a probability distribution $\boldsymbol{p}_t$ over K actions. Nature then reveals a loss vector $\boldsymbol{\ell}_t$ that indicates the loss for each expert. Finally, Learner suffers a loss $\hat{\ell}_t := \boldsymbol{p}_t \cdot \boldsymbol{\ell}_t = \sum_{i=1}^{K} p_{i,t}\ell_{i,t}$.

In the dying experts problem, we assume that the set of experts can only shrink. More formally, for the set of experts $E = \{e_1, e_2, \dots e_K\}$, at each round t, Nature chooses a non-empty set of experts E_a^t to be available such that $E_a^{t+1} \subseteq E_a^t$ for all $t \in \{1, \dots, T-1\}$. In other words, in some rounds Nature sets some experts to sleep, and they will never be available again. Similar to Kleinberg et al. (2010); Kanade et al. (2009); Kanade and Steinke (2014), we adopt the ranking regret as our notion of regret. Before proceeding to the definition of ranking regret, let us define π to be an ordering over the set of initial experts E. We use the notion of orderings and permutation experts interchangeably. Learner can now predict using $\pi \in \Pi$, where Π is the set of all the orderings. Also, denote by $\sigma^t(\pi)$ the first alive expert of ordering π in round t; expert $\sigma^t(\pi)$ is the action that will be played by π. The cumulative loss of an ordering π with respect to the available experts E_a^t is the sum of the losses of

$\sigma^t(\pi)$ at each round. We can now define the ranking regret:

$$R_\Pi(1,T) = \sum_{t=1}^{T} \hat{\ell}_t - \min_{\pi\in\Pi} \sum_{t=1}^{T} \ell_{\sigma^t(\pi),t} \ . \tag{3.1}$$

Since we will use the notion of classical regret in our proofs, with a small change in notation compared to (2.1), we re-define it as:

$$R_E(1,T) = \sum_{t=1}^{T} \hat{\ell}_t - \min_{i\in[K]} \sum_{t=1}^{T} \ell_{i,t} \ . \tag{3.2}$$

We use the convention that the subscript of a regret notion R represents the set of experts against which we compare Learner's performance. Also, the argument in parentheses represents the set of rounds in the game. For example, $R_\Pi(1,T)$ represents the regret over rounds 1 to T with the comparator set being all permutation experts Π. Also, we assume that $\ell_{i,t} \in [0,1]$ for all $i \in [K], t \in [T]$.

Similar to the definition of E_a^t, let $E_d^t := E \setminus E_a^t$ be the set of dead experts at the start of round t. We refer to a round as a "night" if any expert becomes unavailable on the next round. A "day" is defined as a continuous subset of rounds where the subset starts with a round after a night and ends with a night. As an example, if any expert become unavailable at the beginning of round t, we refer to round $t-1$ as a night (and we say the expert dies on that night) and the set of rounds $\{t, t+1 \ldots, t'\}$ as a day, where t' is the next night. We denote by m the number of nights throughout a game of T rounds.

Related work. The papers Freund et al. (1997) and Blum (1997) initiated the line of work on the sleeping experts setting. These works were followed by Blum and Mansour (2007), which considered a different notion of regret and a variety of different assumptions. In Freund et al. (1997), the comparator set is the set of all probability vectors over K experts, while we compare Learner's performance to the performance of the best ordering. In particular, the problem considered in Freund et al. (1997) aims to compare Learner's performance to the best mixture of actions, which also includes our comparator set (orderings). However, in order to recover an ordering as we define, one needs to assign very small probabilities to all experts except for one (the first alive action), which makes the bound in Freund et al. (1997) trivial. As already mentioned, we assume the set E_a^t is chosen adversarially (subject to the

restrictions of the dying setting), while in Kanade et al. (2009) and Neu and Valko (2014) the focus is on the (full) sleeping experts setting with adversarial losses but *stochastic* generation of E_a^t.

For the case of adversarial selection of available actions (which is more relevant to the present paper), Kleinberg et al. (2010) studies the problem in the cases of stochastic and adversarial rewards with both full information and bandit feedback. Among the four settings, the adversarial full-information setting is most related to our work. They prove a lower bound of $\Omega(\sqrt{TK \log K})$ in this case and a matching upper bound by creating $K!$ experts and running Hedge on them, which, as mentioned before, requires computation of order $\mathcal{O}(K^K)$ per round. They prove an upper bound of $\mathcal{O}(K\sqrt{T \log K})$ which is optimal within a log factor for the bandit setting using a similar transformation of experts. A similar framework in the bandits setting introduced in Chakrabarti et al. (2009) is called "mortal bandits"; we do not discuss this work further as the results are not applicable to our case, given that they do not consider adversarial rewards. There is also another line of work which considers the contrary direction of the dying experts game. The setting is usually referred to as "branching" experts, in which the set of experts can only expand. In particular, part of the inspiration for our algorithms came from Gofer et al. (2013); Mourtada and Maillard (2017).

The hardness of the sleeping experts setting is well-studied (Kanade and Steinke (2014); Kale et al. (2016); Kleinberg et al. (2010)). First, Kleinberg et al. (2010) showed for a restricted class of algorithms that there is no efficient no-regret algorithm for sleeping experts setting unless $RP = NP$. Following this, Kanade and Steinke (2014) proved that the existence of a no-regret efficient algorithm for the sleeping experts setting implies the existence of an efficient algorithm for the problem of PAC learning DNFs, a long-standing open problem. For the similar but more general case of online sleeping combinatorial optimization (OSCO) problems, Kale et al. (2016) showed that an efficient and optimal algorithm for "per-action" regret in OSCO problems implies the existence of an efficient algorithm for PAC learning DNFs. Per-action regret is another natural benchmark for partial availability of actions for which the regret with respect to an action is only considered in rounds in which that action was available.

3.3 Number of effective experts in dying experts setting

In this section, we consider the number of effective permutation experts among the set of all possible orderings of initial experts. The idea behind this is that, given the structure in dying experts, not all the orderings will behave uniquely in hindsight. Formally, the *behavior* of π is a sequence of predictions $(\sigma^1(\pi), \sigma^2(\pi), \ldots, \sigma^T(\pi))$. This means that the behaviors of two permutation experts π and π' are the same if they use the same initial experts in *every* round. We define the set of effective orderings $\mathcal{E} \subseteq \Pi$ to be a set such that, for each unique behavior of orderings, there only exists one ordering in $\mathcal{E}$.

To clarify the definition of unique behavior, suppose initial expert e_1 is always awake. Then two orderings $\pi_1 = (e_1, e_2, \ldots)$ and $\pi_2 = (e_1, e_3, \ldots)$ will behave the same over all the rounds, making one of them redundant. Let us clarify that behavior is not defined based on losses, e.g., if $\pi_1 = (e_i, \ldots)$ and $\pi_2 = (e_j, \ldots)$ where $i \neq j$ both suffer identical losses over all the rounds (i.e. their performances are equal) while using different original experts, then they are not considered redundant and hence both of them are said to be *effective*.

Let d_i be the number of experts dying on the i^{th} night. Denote by A the number of experts that will always be awake, so that $A = K - \sum_{i=1}^{m} d_i$. We are now ready to find the cardinality of set $\mathcal{E}$.

Theorem 3 *In the dying experts setting, for K initial experts and m nights, the number of effective orderings in Π is $f(\{d_1, d_2, \ldots d_m\}, A) = A \cdot \prod_{s=1}^{m}(d_s + 1)$.*

Proof We define a new operator denoted by $+$ which operates between an expert e on the left hand side and a multi-set of orderings Π on the right hand side and returns a new multi-set of orderings in which e is added to the left side of every ordering $\pi \in \Pi$. Let $J = \{j_1, \ldots, j_m\}$ be the rounds on which any expert will die.

Without loss of generality, assume that experts die in the order of their indices, i.e., e_1 dies first, e_2 second, ..., and d_{K-A} dies last. We use mathematical induction on m to prove the claim.

Induction Basis: For $m = 0$, or the case that no expert dies (i.e. $A = K$), the number of effective permutation experts is equal to A, the number of experts that never die. Hence,

$$f(\{\}, A) = A.$$

Induction Hypothesis: We assume that the number of effective experts when there are only $i-1$ nights is equal to

$$f\left(\{d_1, d_2, \ldots, d_{i-1}\}, A\right) = A\prod_{s=1}^{i-1}(d_s+1)$$

Denote the set of the effective permutations created in the induction hypothesis as $\mathcal{E}_{i-1}$.

Induction Step: First, notice that any expert $e \in E_d^{j_1+1}$ has an impact on the behavior of a permutation expert π only if $e = \sigma^1(\pi)$. If we ignore the first night and remove every $e \in E_d^{j_1+1}$ from the orderings, the theorem would behave as though those experts do not exist and there are only $i-1$ nights. Due to the induction hypothesis we know that

$$f(\{d_2, \ldots, d_i\}, A) = A\prod_{s=2}^{i}(d_s+1).$$

Denote by F the number of effective orderings π where $e = \sigma^1(\pi)$ for some $e \in E_d^{j_1+1}$. It is easy to see that

$$f(\{d_1, d_2, \ldots, d_i\}, A) = F + f(\{d_2, \ldots, d_i\}, A).$$

On the other hand, the effective orderings which start with $e_s \in E_d^{j_1+1}$ can be constructed as $(e_s) + \mathcal{E}_{i-1}$, so

$$\left|(e_s) + \mathcal{E}_{i-1}\right| = \left|\mathcal{E}_{i-1}\right| = f\left(\{d_2, \ldots, d_i\}, A\right),$$

and it follows that $\mathcal{E}_i = \left(\cup_{e_s \in E_d^{j_1+1}}((e_s) + \mathcal{E}_{i-1})\right) \cup \mathcal{E}_{i-1}$. Then due to the induction hypothesis we get:

$$f(\{d_1, d_2, \ldots, d_i\}, A) = (d_1+1)f(\{d_2, \ldots, d_i\}, A) = (A)\Pi_{s=1}^{i}(d_s+1) \tag{3.3}$$

and (3.3) completes the induction step, concluding the proof. ∎

In the special case where no expert dies ($m = 0$), we use the convention that the (empty) product evaluates to 1 and hence $f(\{\}, A) = A$. We mainly care about $|\mathcal{E}|$ as we use it to derive our upper bounds; hence, we should find the maximum value of f. We can consider the maximum value of f in three regimes.

1. In the case of a fixed number of nights m and fixed A, the function f is maximized

by equally spreading the dying experts across the nights. As the number of dying experts might not be divisible by the number of nights, some of the nights will get one more expert than the others. Formally, the maximum value is $(\lceil \frac{D}{m} \rceil^{D \bmod m} \cdot \lfloor \frac{D}{m} \rfloor^{m-(D \bmod m)} \cdot A)$, where $D = K - A + m$ and $K - A \leq m$.

2. In the case of a fixed number of dying experts (fixed A), the maximum value of f is $(2^{K-A} \cdot A)$ which occurs when one expert dies on each night. The following is a brief explanation on how to get this result. Denote by $B = (d_1, d_2, \ldots, d_b)$ a sequence of numbers of dying experts where more than one expert dies on some night and B maximizes f (for fixed A), so that $F = f(\{d_1, d_2, \ldots, d_b\}, A)$. Without loss of generality, assume that $d_1 > 1$. Split the first night into d_1 days where one expert dies at the end of each day (and consequently each of those days becomes a night). Now $F' = f(\{1, 1, \ldots, 1, d_2, \ldots, d_b\}, A)$ where 1 is repeated d_1 times. If $d_1 > 1$ then $F' = F \cdot 2^{d_1}/(d_1 + 1) > F$. We see that by splitting the nights we can achieve a larger effective set.

3. In the case of a fixed number of nights m, similar to the previous cases, the maximum value is obtained when each night has equal impact on the value of f, i.e., when $A = d_1 + 1 = d_2 + 1 = \cdots = d_m + 1$; however, it might not be possible to distribute the experts in a way to get this, in which case we should make the allocation $\{A, d_1 + 1, d_2 + 1, \ldots, d_m + 1\}$ as uniform as possible.

By looking at cases 2 and 3, we see that by increasing m and the number of dying experts, we can increase f; thus, the maximum value of f with no restriction is 2^{K-1} and is achieved when $m = K - 1$ and $A = 1$.

3.4 Regret bounds for known and unknown order of dying

In this section, we provide lower and upper bounds for the cases of unknown and known order of dying. In order to prove the lower bounds, we need a non-asymptotic minimax lower bound for the DTOL framework, i.e., one which holds for a finite number of experts K and finite T. During the preparation of the final version of this work, we were made aware of a result of Orabona and Pál (see Theorem 8 of Orabona and Pál (2015)) that does give such a bound. However, for completeness, we present a different fully non-asymptotic result that we independently developed; this result is

stated in a simpler form and admits a short proof (though we admit that it builds upon heavy machinery). We then will prove matching upper bounds for both cases of unknown and known order of dying.

3.4.1 Fully non-asymptotic minimax lower bound for DTOL

We analyze lower bounds on the minimax regret in the DTOL game with K experts and T rounds. We assume that all losses are in the interval $[0, 1]$. Let $\Delta_K := \Delta([K])$ denote the simplex over K outcomes. The minimax regret is defined as

$$\inf_{\boldsymbol{p}_1 \in \Delta_K} \sup_{\boldsymbol{\ell}_1 \in [0,1]^K} \dots \inf_{\boldsymbol{p}_T \in \Delta_K} \sup_{\boldsymbol{\ell}_T \in [0,1]^K} \left\{ \sum_{t=1}^{T} \boldsymbol{p}_t \cdot \boldsymbol{\ell}_t - \min_{j \in [K]} \sum_{t=1}^{T} \ell_{j,t} \right\}. \tag{3.4}$$

Theorem 4 *For a universal constant L, the minimax regret (3.4) is lower bounded by*

$$\frac{1}{L} \min \left\{ \sqrt{(T/2) \log K}, T \right\}.$$

The proof begins similarly to the proof of the often-cited Theorem 3.7 of Cesa-Bianchi and Lugosi (2006), but it departs at the stage of lower bounding the Rademacher sum; we accomplish this lower bound by invoking Talagrand's Sudakov minoration for Bernoulli processes (Talagrand (1993, 2005)).

Proof Any strategy of Learner over T rounds can be represented as a sequence $\boldsymbol{p}$ of T maps $\boldsymbol{p}_1, \dots, \boldsymbol{p}_T$, where

$$\boldsymbol{p}_t : [0, 1]^{t-1} \to \Delta_K.$$

By representing Learner's strategy in this way, we can write the above minimax regret as:

$$\inf_{\mathbf{p}} \sup_{\boldsymbol{\ell}_1, \dots, \boldsymbol{\ell}_T} \left\{ \sum_{t=1}^{T} \boldsymbol{p}_t \cdot \boldsymbol{\ell}_t - \min_{j \in [K]} \sum_{t=1}^{T} \ell_{j,t} \right\}.$$

The minimax regret can only decrease by replacing the supremum over the experts' losses by an expectation over random i.i.d. losses $\ell_{j,t}$, where, for all $j \in [K]$ and $t \in [T]$, we take $\ell_{j,t}$ to be independently drawn uniformly from $\{0, 1\}$; hence, the above is

lower bounded by

$$\inf_{\mathbf{p}} \mathsf{E}\left[\sum_{t=1}^{T} \boldsymbol{p}_t \cdot \boldsymbol{\ell}_t - \min_{j\in[K]} \sum_{t=1}^{T} \ell_{j,t}\right] = \mathsf{E}\left[\frac{T}{2} - \min_{j\in[K]} \sum_{t=1}^{T} \ell_{j,t}\right]$$
$$= \frac{1}{2}\,\mathsf{E}\left[\max_{j\in[K]} \sum_{t=1}^{T} (1-2\ell_{j,t})\right].$$

Now, observe that each random variable $(1-2\ell_{j,t})$ has the same law as an independent Rademacher random variable (i.e. uniform over ± 1). Therefore, the above is equal to

$$\frac{1}{2}\,\mathsf{E}\left[\max_{j\in[K]} \sum_{t=1}^{T} \varepsilon_{j,t}\right], \tag{3.5}$$

where the $\varepsilon_{j,t}$ are independent Rademacher random variables.

Our approach will be to express the above as the expected maximum of a Bernoulli process. To this end, for each $j \in [K]$ define the matrix $\tau^{(j)} \in \mathbb{R}^{T\times K}$ whose j^{th} column is equal to the ones vector and whose remaining columns each are equal to the zero vector. Our lower bound on the minimax regret now can be rewritten as (one half of)

$$\mathsf{E}\left[\max_{j\in[K]} \sum_{t=1}^{T} \varepsilon_{j,t}\right] = \mathsf{E}\left[\max_{j\in[K]} \sum_{i=1}^{K}\sum_{t=1}^{T} \varepsilon_{i,t}\tau_{i,t}^{(j)}\right].$$

This is just the expected supremum of a Bernoulli process indexed by $\{\tau^{(1)}, \ldots, \tau^{(K)}\}$. A result of Talagrand (2005), restated as Lemma 2 after this proof, can be used to lower bound this process in terms of T and K. In our setting, treating the matrices $\tau^{(j)}$ as vectors by stacking their columns, we see that the vectors $(\tau^{(j)})_j$ satisfy

- $\|\tau^{(i)} - \tau^{(j)}\|_2 \geq \sqrt{2T}$ for all distinct $i, j \in [K]$;
- $\|\tau^{(j)}\|_\infty \leq 1$.

Hence, Lemma 2 implies that the minimax regret is lower bounded by

$$\frac{1}{2L} \min\left\{\sqrt{2T\log K}, 2T\right\},$$

for L a universal constant. ∎

The above proof uses the following powerful result of Talagrand on Sudakov

minoration for Bernoulli processes; here, the most convenient form is stated as Theorem 4.2.4 in Talagrand (2005), but the result first appeared as Proposition 2.2 of Talagrand (1993) in a quite different form.

Lemma 2 (Sudakov minoration for Bernoulli processes) *Let $a, b > 0$ and $\tau^{(1)}, \ldots, \tau^{(K)} \in \ell^2$ satisfy the conditions:*

- $\|\tau^{(i)} - \tau^{(j)}\|_2 \geq a$ *for all distinct* $i, j \in [K]$;
- $\|\tau^{(j)}\|_\infty \leq b$ *for all* $j \in [K]$,

Let $\varepsilon_1, \varepsilon_2, \ldots$ be i.i.d Rademacher random variables.

Then for a universal constant L we have

$$\mathsf{E} \sup_{j \leq K} \sum_{s \geq 1} \tau_s^{(j)} \varepsilon_s \geq \frac{1}{L} \min \left\{ a\sqrt{\log K}, \frac{a^2}{b} \right\}.$$

3.4.2 Unknown order of dying

For the case where Learner is not aware of the order in which the experts die, we prove a lower bound of $\Omega(\sqrt{mT \log K})$. Given that we have $E_a^{t+1} \subseteq E_a^t$, the construction for the lower bound of Kleinberg et al. (2010) cannot be applied to our case. In other words, our adversary is much weaker than the one in Kleinberg et al. (2010), but, surprisingly, we show that the previous lower bound still holds (by setting $m = K$) even with the weaker adversary. We then analyze a simple strategy to achieve a matching upper bound.

In this section, we further assume that $\sqrt{T/2 \log K} < T$ for every T and K so that there is hope to achieve regret that is sublinear with respect to T. We now present our lower bound on the regret for the case of unknown order of dying.

Theorem 5 *When the order of dying is unknown, the minimax regret is* $\Omega(\sqrt{mT \log K})$.

Proof We prove the theorem using a construction as follows. Recall that we refer to a round as a "night" if an expert dies on that round and to each segment between two nights as a "day". First, partition T rounds to rounds of length $T' = T/(m+1)$, where m is the number of nights. The goal is to construct a scenario where each day is a game decoupled from the previous ones. This means that the algorithm will be forced to have no prior information about the experts at the beginning of each day. Recall that τ_s is the set of time-step indices of day s, i.e.,$\tau_s = \{t | (s-1)T' < t \leq sT'\}$.

Each day is divided into two equal parts. Denote by τ_s^1 and τ_s^2 sets of time-step indices of the first half and the second half of day s, respectively. Let $\boldsymbol{\ell}_{\tau_s^1,i}$ and $\boldsymbol{\ell}_{\tau_s^2,i}$ be the sequences of losses of an expert i on the first and second half of the day s, respectively. On the first half of day s, each expert suffers loss drawn i.i.d. from a Bernoulli distribution with $p = 1/2$. At the end of the first half of the day, we choose the expert with the lowest cumulative loss up until now, denoted by e_s^*. This expert will suffer no loss in the second half. Also, the adversary forces every other expert i where $e_i \neq e_s^*$ to suffer losses according to loss sequence $\boldsymbol{\ell}_{s_2,i}$ on the second half of the day, where we have element-wise subtraction as $\boldsymbol{\ell}_{s_2,i} = \mathbf{1} - \boldsymbol{\ell}_{s_1,i}$. Denote by $E_a(s)$ the set of experts alive on day s.

We now analyze the ranking regret of any algorithm for this construction over T rounds. Without loss of generality, suppose the order of experts that are going to die is as $\mathcal{D} = (e_1, e_2, \ldots, e_m)$. Also, denote by $\pi^* \in \Pi$ the best ordering over T. From the construction, it is clear that $\pi^* = (\mathcal{D}, \ldots)$. Therefore, the ranking regret over T rounds is obtained from (3.6).

$$R_\Pi(1,T) = \hat{L} - L_{\pi^*} = \hat{L} - \sum_{s=1}^{m+1} \sum_{t \in \tau_s} l_{\sigma^t(\pi^*),t} \tag{3.6}$$

where L_{π^*} is the cumulative loss of playing according to ordering π^*. Now we write $R_\Pi(1,T)$ in terms of a sum of classical regrets over each day. Since in our construction the best expert of the day will die at the end of that day, then, for all rounds in a given day, $\sigma^t(\pi^*)$ yields the same expert as the expert that is best for that day according to the ordinary regret. Therefore, we have:

$$\begin{aligned} \hat{L} - \sum_{s=1}^{m+1} \sum_{t \in \tau_s} l_{\sigma^t(\pi^*),t} &= \sum_{s=1}^{m+1} \left(\sum_{t \in \tau_s} \hat{l}_t - \min_{s \leq i \leq K} \sum_{t \in \tau_s} l_{i,t} \right) \\ &= \sum_{s=1}^{m+1} R_{E_a(s)}(\tau_s) \end{aligned} \tag{3.7}$$

where the last equality is obtained from the fact that in our construction, each day is an independent day from the others, meaning the history of losses of the experts does not matter. Combining (3.7) and (3.6), we have:

$$R_\Pi(1,T) = \sum_{s=1}^{m+1} R_{E_a(s)}\left(\tau_s\right) \tag{3.8}$$

Now it remains to analyze the regret of each day separately. For this, first, we lower bound the regret of each half of the days. Denote by $\hat{L}_s^1$ and $\hat{L}_s^2$ the cumulative losses of the algorithm on the first and second half of the day s with length, respectively. It is easy to verify that $R_{E_a(s)}(\tau_s) \geq R_{E_a(s)}(\tau_s^2)$, hence for the regret of each day we have

$$\begin{aligned} R_{E_a(s)}(\tau_s) = \hat{L}_s^1 + \hat{L}_s^2 - \sum_{t\in\tau_s} l_{\sigma^t(\pi^*),t} &\geq \hat{L}_s^1 - \sum_{t\in\tau_s^1} l_{\sigma^t(\pi^*),t} \\ &\geq \frac{1}{L}\min\{\sqrt{T'/2\log(K-s)}, T'\} \end{aligned} \tag{3.9}$$

where the last inequality is based on (the proof of) Theorem 4. Combining (3.8) and (3.9) we have:

$$\begin{aligned} R_{\Pi}(1,T) &\geq \sum_{s=1}^{m+1} \frac{1}{L}\min\{\sqrt{T'/2\log(K-s)}, T'\} \\ &= \sum_{s=1}^{m+1} \sqrt{T/2(m+1)\log(K-s)} = \Omega\left(\sqrt{Tm\log K}\right), \end{aligned} \tag{3.10}$$

yielding the desired bound. ∎

The proof of Theorem 5 uses the results of following Lemma.

Lemma 3 *For variables $x_1, \ldots, x_m > 0$ and subject to $\sum_{i=1}^m x_i = T$, we have:*

$$\sum_{i=1}^{m} \sqrt{x_i} \leq \sqrt{mT}$$

PROOF Denote by $\mathcal{T}$ the vector $\left[\sqrt{x_1}, \sqrt{x_2}, \ldots, \sqrt{x_m}\right]$ and $I = [1, 1, \ldots, 1]$ vectors of length m where all the elements are equal to one. We have:

$$\sum_{i=1}^{m} \sqrt{x_i} = \mathcal{T}\cdot I^T \leq ||\mathcal{T}||\cdot||I|| \leq \sqrt{m}\sqrt{(\sqrt{x_1})^2 + (\sqrt{x_2})^2 + \cdots + (\sqrt{x_m})^2} \leq \sqrt{mT}$$

where the first inequality follows from the Cauchy-Schwarz inequality. ∎

A natural strategy in the case of unknown dying order is to run Hedge over the set of initial experts E and, after each night, reset the algorithm. We will refer to this strategy as "Resetting-Hedge". Theorem 6 gives an upper bound on regret of Resetting-Hedge.

Theorem 6 *Resetting-Hedge enjoys a regret of* $R_{\Pi}(1,T) = \mathcal{O}(\sqrt{mT \log K})$.

PROOF Let τ_s be the set of round indices of day s; hence, we have $\sum_{s=1}^{m+1} |\tau_s| = T$. The overall ranking regret can be upper bounded by the sum of classical regrets for every interval. Hence, the analysis is as follows:

$$R_{\Pi}(1,T) \leq \sum_{s=1}^{m+1} \sqrt{|\tau_s| \log(K-s)} \leq \sqrt{\log K} \sum_{s=1}^{m+1} \sqrt{|\tau_s|} \leq \sqrt{(m+1)T \log K}; \quad (3.11)$$

the last inequality is essentially from the Cauchy-Schwarz inequality (see Lemma 3).■

Although the basic Resetting-Hedge strategy adapts to m, it has many downsides. For example, resetting can be wasteful in practice. Another natural strategy, simply running Hedge on the set of all $K!$ permutation experts, is non-adaptive (obtaining regret $\mathcal{O}(\sqrt{TK \log K})$ and computationally inefficient if implemented naïvely). However, as we show in Section 3.5.1, this algorithm can be implemented efficiently (with runtime linear in K rather than $K!$) and also, as we show in Section 3.5.3, by running Hedge on top of several copies of Hedge (one per specially chosen learning rate), we can obtain a guarantee that is far better than Theorem 6. Moreover, our efficient implementation of Hedge can be extended to adaptive algorithms like AdaHedge and FlipFlop (de Rooij et al. (2014)).

3.4.3 Known order of dying

A natural question is whether Learner can leverage information about the order of experts that are going to die to achieve a better regret. We show that the answer is positive: the bound can be improved by a logarithmic factor. We also give a matching lower bound for this case (so both bounds are tight).

Similar to the unknown setting, we provide a construction to prove a lower bound on the ranking regret in this case. We still assume that $\sqrt{T/2 \log K} < T$.

Theorem 7 *When Learner knows the order of dying, the minimax regret is* $\Omega(\sqrt{mT})$.

PROOF The construction for this case is similar to the one we previously had for the unknown order of dying. We divide the T rounds into $m/2$ days each of size $T' = 2T/m$. We choose two experts $\{e_{2s-1}, e_{2s}\}$ at each day s and they will suffer losses drawn i.i.d. from a Bernoulli distribution with success probability of $p = 1/2$. Every expert $e_i \notin \{e_{2s-1}, e_{2s}\}$ will suffer constant loss of 1 during day s. At the end of

day s, both the experts $\{e_{2s-1}, e_{2s}\}$ will die. Therefore, at the beginning of each day, all the experts have the same loss history and consequently, each day is decoupled from the previous ones. Additionally, we provide extra information to the algorithm, that the best expert of day s is one of the two experts $\{e_{2s-1}, e_{2s}\}$. Thus, the algorithm needs to track only two experts on a single day.

Denote by $E_a(s)$ the set of experts alive on day s. In the following, we analyze the ranking regret of this construction. We will use the same result from the proof of Theorem 5 to connect ranking regret to the classical regret over each day. Hence, using (3.8), we have:

$$R_\Pi(1,T) = \sum_{s=1}^{m/2} R_{E_a(s)}(\tau_s) \tag{3.12}$$

Using the bound we obtained from Theorem 4, for each day s, $K = 2$ and T' rounds we have:

$$R_{E_a(s)}(\tau_s) \geq \frac{1}{L} \min\{\sqrt{T'/2\log 2}, T'\} \tag{3.13}$$

Combining (3.12) and (3.13), we obtain the bound on ranking regret over time horizon T as follows:

$$R_\Pi(1,T) \geq \sum_{s=1}^{m/2} \frac{1}{L} \min\{\sqrt{T'/2\log 2}, T'\} = \sum_{s=1}^{m/2} \sqrt{T/m} = \Omega\left(\sqrt{mT}\right)$$

The theorem follows. ∎

Although the proof is relatively simple, it is at least a little surprising that knowing such rich information as the order of dying only improves the regret by a logarithmic factor.

To achieve an optimal upper bound, using the results of Theorem 3, the strategy is to create $2^m(K-m)$ experts (those that are effective) and run Hedge on this set.

Theorem 8 *For the case of known order of dying, the strategy as described above achieves a regret of* $\mathcal{O}(\sqrt{T(m+\log K)})$.

Proof Hedge has regret of $\mathcal{O}(\sqrt{T\log K})$ for K number of experts. Therefore, running Hedge on $2^m(K-m)$ experts yields the desired bound. ∎

Though the order of computation in the above strategy is better than $\mathcal{O}(K^K)$, it is still exponential in K. In the next section, we introduce algorithms that simulate these strategies but in a computationally efficient way.

3.5 Efficient algorithms for dying experts

The results of Kanade and Steinke (2014) imply computational hardness of achieving no-regret algorithms in sleeping experts; yet, we are able to provide efficient algorithms for dying experts in the cases of unknown and known order of dying. For the sake of simplicity, we initially assume that only one expert dies each night. Later, in Section 3.5.3, we show how to extend the algorithms for the general case where multiple experts can die each night. We then show how to extend these algorithms to adaptive algorithms such as AdaHedge (de Rooij et al. (2014)). The algorithms for both cases are given in Algorithms 2 and 3.

3.5.1 Unknown order of dying

We now show how to efficiently implement Hedge over the set of all the orderings. Even though Resetting-Hedge is already efficient and achieves optimal regret, it has its own disadvantages. The issue arises when one needs to extend Resetting-Hedge to adaptive algorithms. This is particularly important in real-world scenarios, where Learner wants to adapt to the environment (such as stochastic or adversarial losses). We show that Algorithm 2, Hedge-Perm-Unknown (HPU), can be adapted to AdaHedge (van Erven et al. (2011)) and, therefore, we can simulate FlipFlop (de Rooij et al. (2014)). Next, we give the main idea on how the algorithm works, after which we prove that Algorithm 2 efficiently simulates running Hedge over Π. Before proceeding further, let us recall how Hedge makes predictions in round t. First, it updates the weights using $w_{i,t} = w_{i,t-1}e^{-\eta\ell_{i,t}}$, and it then assigns a probability to expert i as follows:

$$p_{i,t} = \frac{w_{i,t-1}}{\sum_{i=1}^{K} w_{i,t-1}} \ .$$

Recall that $e_1, e_2, \ldots, e_K$ denote the original experts while $\pi_1, \pi_2, \ldots \pi_{K!}$ denote the orderings. Denote by w_π^t the weight that Hedge assigns to π in round t. Define $\Pi_i^t \subseteq \Pi$ to be the set of orderings predicting as expert e_i in round t. The main ideas behind the algorithm are as follows:

1. When π and π' have the same prediction e in round t (i.e. $\sigma^t(\pi) = \sigma^t(\pi') = e$), then we do not need to know w_π^t and $w_{\pi'}^t$; we use $w_\pi^t + w_{\pi'}^t$ instead for the weight of e.

2. The algorithm maintains $\sum_{\pi\in\Pi_j^t} e^{-\eta L_\pi^{t-1}}$, where η is the learning rate and L_π^t is

the cumulative loss of ordering π up until round t, i.e., $L_\pi^t = \sum_{s=1}^t \ell_{\sigma^s(\pi),s}$.

Algorithm 2: Hedge-Perm-Unknown (HPU)

$\forall i \in [K]$ $c_{i,1} := 1$, $h_{i,1} := (K-1)!$, $E_a := \{e_1, e_2, \dots e_K\}$
for *t = 1, 2, ..., T* **do**
  play $\mathbf{p}_t = \left[\mathbf{1}\left[e_i \in E_a\right] \cdot \frac{h_{i,t} \cdot c_{i,t}}{\sum_{j=1}^k h_{j,t} \cdot c_{j,t}}\right]_{i \in [K]}$
  receive $(\ell_{1,t}, \dots, \ell_{K,t})$
  for $e_i \in E_a$ **do**
    $c_{i,t+1} := c_{i,t} \cdot e^{-\eta \ell_{i,t}}$
    $h_{i,t+1} := h_{i,t}$
  if *expert j dies* **then**
    $E_a := E_a \setminus \{e_j\}$
    for $e_i \in E_a$ **do**
      $h_{i,t+1} := h_{i,t+1} \cdot c_{i,t+1} + (h_{j,t+1} \cdot c_{j,t+1})/|E_a|$
      $c_{i,t+1} := 1$

We will discuss how to tune η later. Let $J = \{j_1, \dots, j_m\}$ represent the rounds on which any expert will die. Denote by j_t the last night observed so far at the end of round t, formally defined as $j_t = \max_{j \in J} j \leq t$. We maintain a tuple $(h_{i,t}, c_{i,t})$ for each original expert e_i's in the algorithm in round t, where $h_{i,t}$ is the sum of non-normalized weights of the experts in Π_i^t in round j_t. We similarly maintain $c_{i,t}$, except that it only considers the loss suffered from $j_t + 1$ to round $t - 1$ for experts in Π_i^t. Formally:

$$h_{i,t} = \sum_{\pi \in \Pi_i^t} e^{-\eta(\sum_{s=1}^{j_t} \ell_{\sigma^s(\pi),s})}, \qquad c_{i,t} = \sum_{\pi \in \Pi_i^t} e^{-\eta(\sum_{s=j_t+1}^{t-1} \ell_{\sigma^s(\pi),s})} .$$

It is easy to verify that $h_{j,t} \cdot c_{j,t} = \sum_{\pi \in \Pi_j^t} e^{-\eta L_\pi^{t-1}}$.

The computational cost of the algorithm at each round will be $\mathcal{O}(K)$. We claim that HPU will behave the same as executing Hedge on Π. We use induction on rounds to show the weights are the same in both algorithms. By "simulating" we mean that the weights over the original experts will be maintained identically to how Hedge maintains them.

Theorem 9 *At every round, HPU simulates running Hedge on the set of experts* Π.

PROOF The main idea is to group the permutation experts with similar predictions (the first expert alive in the permutation) in one group. Hence, initially there will be

K groups. Then, if expert e_j dies, every ordering in the group associated with e_j will be moved to another group and the empty group will be deleted. We prove that the orderings will distribute to other groups symmetrically after a night. Using this fact, we show that we do not need to know the elements of a group; we only maintain the sum of the weights given to all the orderings in each group.

We show that the loss and weights of the algorithms are the same at each round, therefore, their regret is the same. Define Π_D to be the set of all possible orderings of elements in set D of length $|D|$. We claim that based on the update rules of HPU for $h_{j,t}$ and $c_{j,t}$, for every round and expert we have $\sum_{\pi\in\Pi_j^t} e^{-\eta L_\pi^{t-1}} = h_{j,t}\cdot c_{j,t}$.

Induction Basis: At round $t = 1$, in Hedge, every expert has non-normalized weight of 1. The size of each $\Pi_{e_j}^1$ is $(K-1)!$. The algorithm assigns $h_{i,1} = (K-1)!$ and $c_{i,1} = 1$, therefore the claims hold.

Induction Hypothesis: At the beginning of round $t-1$, for every alive expert e_j, the following holds:

$$\sum_{\pi\in\Pi_j^{t-1}} e^{-\eta L_\pi^{t-2}} = h_{j,t-1}\cdot c_{j,t-1} \tag{3.14}$$

Induction Step: This step is divided into two cases. First, when $E_a^t = E_a^{t-1}$. Second, when an expert dies, $|E_a^t| = |E_a^{t-1}| - 1$.

Case I: If no expert dies at the end of round $t-1$, then for every i we have $\Pi_i^t = \Pi_i^{t-1}$ and $h_{i,t} = h_{i,t-1}$, thus for every alive expert e_j following holds $\sum_{\pi\in\Pi_j^t} e^{-\eta L_\pi^{t-2}} = h_{j,t-1}\cdot c_{j,t-1}$. After the update in Hedge, the weights on Π_j^t is $\sum_{\pi\in\Pi_j^t} e^{-\eta(L_\pi^{t-2}+\ell_{j,t-1})}$. On the other hand, in the HPU algorithm, we have the following:

$$\begin{aligned} h_{j,t}\cdot c_{j,t} = e^{-\eta\ell_{j,t-1}}\cdot c_{j,t-1}\cdot h_{j,t-1} &= e^{-\eta\ell_{j,t-1}} \sum_{\pi\in\Pi_j^{t-1}} e^{-\eta L_\pi^{t-2}} \\ &= \sum_{\pi\in\Pi_j^t} e^{-\eta(L_\pi^{t-2}+\ell_{j,t-1})} = \sum_{\pi\in\Pi_j^t} e^{-\eta L_\pi^{t-1}} \end{aligned}$$

Where the second equality follows from the induction hypothesis. It can be observed that the weights are identical to the ones from running Hedge on $K!$ experts.

Case II: The second case is when the expert j dies at the end of round $t-1$. Let i, k be arbitrary alive experts not equal to j. Observe that any $\pi \in \Pi_i^t \cap \Pi_j^{t-1}$ takes the form $(\pi_d, e_j, \pi_{d'}, e_i, \pi_{R_i})$, where, for some $D, D' \subseteq E_d^t$ where $D \cap D' = \emptyset$ and $R_i := E \setminus (D \cup D' \cup \{e_j, e_i\})$, we have that $\pi_d \in \Pi_D$ and $\pi_{d'} \in \Pi'_D$ and $\pi_{R_i} \in \Pi_{R_i}$. Then $\Pi_k^t \cap \Pi_j^{t-1}$ contains a unique element $\pi' = (\pi_d, e_j, \pi_{d'}, e_k, \pi_{R_k})$, where D is taken

as before and (like before) $R_k := E \setminus (D \cup D' \cup \{e_j, e_k\})$ and π_{R_k} is created only by replacing e_i as e_k in π_{R_i}. Moreover, since their behavior is the same over the first $t-1$ rounds, π and π' satisfy $L_\pi^{t-1} = L_{\pi'}^{t-1}$.

Therefore by symmetry, we can obtain (3.16) from (3.15).

$$\sum_{\pi \in \Pi_i^t} e^{-\eta L_\pi^{t-1}} = \sum_{\pi \in \Pi_i^{t-1}} e^{-\eta L_\pi^{t-1}} + \sum_{\pi \in (\Pi_i^t \bigcap \Pi_j^{t-1})} e^{-\eta L_\pi^{t-1}} \tag{3.15}$$

$$= \sum_{\pi \in \Pi_i^{t-1}} e^{-\eta L_\pi^{t-1}} + \frac{1}{|E_a^t|} \left(\sum_{\pi \in \Pi_j^{t-1}} e^{-\eta L_\pi^{t-1}} \right) \tag{3.16}$$

$$= h_{i,t} \cdot c_{i,t}$$

Notice that, given expert j dies at the end of round $t-1$ hence, $\sum_{\pi \in \Pi_j^{t-1}} e^{-\eta L_\pi^{t-1}}$ is calculable. Therefore, HPU is always maintaining the weights correctly. ∎

3.5.2 Known order of dying

For the case of known order of dying, we propose Algorithm 3, Hedge-Perm-Known (HPK), which is slightly different than HPU. In particular, the weight redistribution (when an expert dies) and initialization of coefficient $h_{i,1}$ is different. In the proof of Theorem 9, we showed that when the set of experts includes all the orderings, the weight of the expert j that died will distribute equally between initial experts ($e_j \in E$). But when the set of experts is only the effective experts, this no longer holds. In this section, we assume without loss of generality that the experts die in the order $e_1, e_2, \ldots$ and recall that $\mathcal{E}$ denotes the set of effective orderings. Based on Theorem 3, the number of experts starting with e_i in $\mathcal{E}$ is $\lceil 2^{K-i-1} \rceil$; we denote the set of such

experts as $\mathcal{E}_{e_i}$.

Algorithm 3: Hedge-Perm-Known (HPK)

$\forall i \in [K]\ c_{i,1} := 1,\ h_{i,1} := \lceil 2^{K-i-1} \rceil,\ E_a := \{e_1, e_2, \dots e_K\}$
for $t = 1, 2, \dots, T$ **do**
  play $\mathbf{p}_t = \left[\mathbf{1}\left[e_i \in E_a\right] \cdot \frac{h_{i,t} \cdot c_{i,t}}{\sum_{j=1}^{k} h_{j,t} \cdot c_{j,t}}\right]_{i \in [K]}$
  receive $(\ell_{1,t}, \dots, \ell_{K,t})$
  for $e_i \in E_a$ **do**
    $c_{i,t+1} := c_{i,t} \cdot e^{-\eta \ell_{i,t}}$
    $h_{i,t+1} := h_{i,t}$
  if *expert j dies* **then**
    $E_a := E_a \setminus \{e_j\}$
    for *each $i = j+1$ to K* **do**
      $h_{i,t+1} := h_{i,t+1} \cdot c_{i,t+1} + (h_{j,t+1} \cdot c_{j,t+1})(\lceil 2^{i-2} \rceil / 2^{K-1-j})$
      $c_{i,t+1} := 1$

Theorem 10 *At every round, HPK simulates running Hedge on the set of experts $\mathcal{E}$.*

Before proceeding to the proof, let us mention that wherever we refer to Theorem 3 in this part, we assume that only one expert dies each night; therefore for m nights (consequently, m dying experts) the value of f (the function defined in Theorem 3) is $2^m(K-m)$ where K is the number of experts.

Proof Here we follow a construction similar to the proof of Theorem 9, i.e., we do induction on t. Before proceeding to the proof, define $\lambda(\pi, t)$ as a function that will remove ineffective elements of a permutation expert at round t. An element is said to be ineffective, if it will never be used for the prediction in that permutation or it is dead. Recall that in this section we assumed that the experts die in order, e_1 dies first and e_{K-A} last. For example, $(e_4) = \lambda((e_4, e_3, e_2, e_1), t)$ and $(e_1, e_3, e_4) = \lambda((e_1, e_3, e_2, e_4), 1)$ with respect to the assumption we made earlier on the order of dying, and if e_1 dies at $t = 1$ and e_3 dies at $t = 5$, then $(e_3, e_4) = \lambda((e_1, e_3, e_2, e_4), 3)$ and $(e_4) = \lambda((e_1, e_3, e_2, e_4), 6)$. Naturally, $\lambda(\mathcal{E}, t)$ performs the function $\lambda(\pi, t)$ on every permutation $\pi \in \mathcal{E}$. The output of $\lambda(\mathcal{E}, t)$ is a multi-set, not a set.

Induction Basis: At round $t = 0$, each of the permutation-experts have the same weight. Due to the Theorem 3, we know the number of the orderings starting by expert e_i is equal to $\lceil 2^{K-i-1} \rceil$. Therefore, in Hedge, the cumulative non-normalized weight put on $\mathcal{E}^1_{e_i}$ is $\lceil 2^{K-i-1} \rceil$ which is equal to $h_{i,1} \cdot c_{i,1}$ in HPK.

Induction Hypothesis: At the beginning of round $t-1$, in Hedge, the cumulative non-normalized weight put on $\mathcal{E}^{t-1}_{e_i}$ for every $e_i \in E^{t-1}_a$, is equal to $h_{i,t-1} \cdot c_{i,t-1}$

Induction Step: As in the proof of Theorem 9, for round t, we split the step into two cases. In the first case, no expert dies, i.e., $E^t_a = E^{t-1}_a$. For the second case, one expert dies at the end of round $t-1$. The proof for the first case is omitted as it is identical to the proof for *Case I* of Theorem 9. For the case that an expert dies at the end of round $t-1$, we show that the weight distribution works correctly.

Let $E^{(i+1,K)} = \{e_{i+1}, \ldots, e_K\}$. Due to the discussions in Section 3.3, first, we have the number of initial orderings starting by e_i as $\lceil 2^{K-i-1} \rceil$ at the beginning. Second, due to Lemma 4, if e_i dies at round $t-1$, we have:

$$\lambda\left(\mathcal{E}^{t-1}_{e_i}, \quad t\right) = \lambda\left(\bigcup_{e \in E^{(i+1,K)}} \mathcal{E}^{t-1}_e, \quad t\right) \tag{3.17}$$

therefore, for every e_j where $j > i$, $(|\mathcal{E}^1_{e_j}|)/(|\mathcal{E}^1_{e_i}|)$ fraction of $h_{i,t-1} \cdot c_{i,t-1}$ must be added to the weight of $\mathcal{E}^t_{e_j}$ to maintain the weight of $\mathcal{E}^t_{e_j}$. ∎

Before proceeding to Lemma 4, recall that operator $+$ operates between an expert e on LHS and a multi-set of orderings Π on RHS and returns a new multi-set of orderings which e is added to the left side of every ordering $\pi \in \Pi$.

Lemma 4 *At round t, where $E^t_d = \{e_1, e_2, \ldots e_{i-1}\}$ are dead and the rest of the experts are alive, we have:*

$$\lambda\left(\mathcal{E}^t_{e_i}, \quad t\right) = (e_i) + \lambda\left(\bigcup_{e \in E^{(i+1,K)}} \mathcal{E}^t_e, \quad t\right)$$

and therefore:

$$\left|\mathcal{E}^t_{e_i}\right| = \left|\lambda\left(\bigcup_{e \in E^{(i+1,K)}} \mathcal{E}^t_e, \quad t\right)\right|.$$

Before proving the statement, let us define two new operators. For $\mathcal{E}$ as the set of permutation-experts, $\mathcal{E} - \{e_i\}$ removes element e_i from every permutation $\pi \in \mathcal{E}$. Also, $\mathcal{E}' = x\mathcal{E}$ is a multi-set where each item in $\mathcal{E}$ is copied x times. As a result, trivially we have $|\mathcal{E}'| = x \cdot |\mathcal{E}|$.

Proof Recall that we assumed that the experts die in order. Due to constructive structure of the Theorem 3, $\mathcal{E}^1_{e_i}$ is equal to adding e_i as the first element for every permutation in $\mathcal{E}^1_{E^{(i+1,K)}}$.

$$\lambda\left((\mathcal{E}^1_{e_i}), \quad 1\right) = (e_i) + \lambda\left(\bigcup_{e \in E^{(i+1,K)}} \mathcal{E}^1_e, \quad 1\right)$$

Therefore, the claim holds for $t = 1$ and we have $|\lambda(\mathcal{E}^1_{e_i}, 1)| = |\mathcal{E}^1_{e_i}|$. It is easy to verify that:

$$\left|\lambda\left(\bigcup_{e\in E^{(i+1,K)}}\mathcal{E}^1_e,\quad 1\right)\right| = \left|\bigcup_{e\in E^{(i+1,K)}}\mathcal{E}^1_e\right|$$

Due to Lemma 5, similar claim holds for t when $\{e_1, \ldots, e_{i-1}\}$ are dead. $\lambda(\mathcal{E}^t_{e_i}, t)$ will be 2^{i-1} copies of $\lambda(\mathcal{E}^1_{e_i}, 1)$ and similarly

$$\lambda\left(\bigcup_{e\in E^{(i+1,K)}}\mathcal{E}^t_e,\quad t\right) = 2^{i-1}\lambda\left(\bigcup_{e\in E^{(i+1,K)}}\mathcal{E}^1_e,\quad 1\right)$$

hence:

$$\begin{aligned} 2^{i-1}\lambda\left(\mathcal{E}^1_{e_i},\quad 1\right) &= (e_i) + 2^{i-1}\lambda\left(\bigcup_{e\in E^{(i+1,K)}}\mathcal{E}^1_e,\quad 1\right)\\ \lambda\left(\mathcal{E}^t_{e_i},\quad t\right) &= (e_i) + \lambda\left(\bigcup_{e\in E^{(i+1,K)}}\mathcal{E}^t_e,\quad t\right) \end{aligned}$$

■

Lemma 5 *At round t when m experts have died so far and $e_j \in E^t_a$, $\lambda(\mathcal{E}^t_{e_j}, t)$ is equal to 2^m copies of $\lambda(\mathcal{E}^1_{e_j}, 1)$.*

Recall that Π_D is the set of all possible orderings of elements in set D of length $|D|$ and similarly, $\mathcal{E}_D$ is the set of all effective orderings with respect to Π_D.

PROOF Due to the constructive building of $\mathcal{E}^1_{e_i}$, $\lambda(\mathcal{E}^1_{e_i})$ is equal to

$$(e_i) + \lambda\left(\mathcal{E}^1_{\{e_{i+1},\ldots,e_K\}},\quad 1\right) = (e_i) + \lambda\left(\bigcup_{i+1\le j\le K}\mathcal{E}^1_{e_j},\quad 1\right) = (e_i) + \bigcup_{i+1\le j\le K}\lambda\left(\mathcal{E}^1_{e_j},\quad 1\right) \tag{3.18}$$

We use induction on m to prove the claim.

Induction Basis: The claim trivially holds when $m = 0$.

Induction Hypothesis: When $e_1, e_2, \ldots, e_{i-1}$ are dead before round $t-1$, for any $j \ge i$ we have $\lambda(\mathcal{E}^{t-1}_{e_j},\quad t-1) = 2^{i-1}\lambda(\mathcal{E}^1_{e_j},\quad 1)$

Induction Step: Assume that at round $t-1$, e_i dies.

$$\begin{aligned} \lambda\left(\mathcal{E}^1_{e_i},\quad 1\right) &= (e_i) + \bigcup_{e\in E^{(i+1,K)}}\lambda\left(\mathcal{E}^1_e,\quad 1\right)\\ 2^{i-1}\lambda\left(\mathcal{E}^1_{e_i},\quad 1\right) &= (e_i) + 2^{i-1}\bigcup_{e\in E^{(i+1,K)}}\lambda\left(\mathcal{E}^1_e,\quad 1\right)\\ 2^{i-1}\lambda\left(\mathcal{E}^1_{e_i},\quad 1\right) &= (e_i) + \bigcup_{e\in E^{(i+1,K)}}2^{i-1}\lambda\left(\mathcal{E}^1_e,\quad 1\right)\\ \lambda\left(\mathcal{E}^{t-1}_{e_i},\quad t-1\right) &= (e_i) + \bigcup_{e\in E^{(i+1,K)}}\lambda\left(\mathcal{E}^{t-1}_e,\quad t-1\right) \end{aligned} \tag{3.19}$$

Where the second and forth equality follows by applying the induction hypothesis to the left and right sides of the first and third line and first equality holds due to

Section 3.3. Therefore when e_i dies, any $\pi \in \mathcal{E}_{e_i}^{t-1}$ we have $\pi \in \mathcal{E}_e^t$ where $e \in E^{(i+1,K)}$ hence

$$\bigcup_{e\in E^{(i+1,K)}} \lambda\left(\mathcal{E}_e^t,\ \ t\right) = \bigcup_{e\in E^{(i+1,K)}} 2\lambda\left(\mathcal{E}_e^{t-1},\ \ t-1\right) = \bigcup_{e\in E^{(i+1,K)}} 2^i\lambda\left(\mathcal{E}_e^1,\ \ 1\right)$$

where the second equality is from the induction hypothesis. This is easy to see that each set in the union is independent from others, so $\lambda(\mathcal{E}_{e_j}^t,\ \ t) = 2^i\lambda(\mathcal{E}_{e_j}^1,\ \ 1)$ where $j > i$. ∎

Remarks for tuning learning rates. For both algorithms, we assume T is known beforehand. So, the learning rate for HPU is $\eta = \sqrt{(2\log(K!))/T}$ and for HPK is $\eta = \sqrt{(2\log(2^m(K-m)))/T}$. One can use a time-varying learning rate to adapt to T in case it is not known.

3.5.3 Extensions for algorithms

As we mentioned at the beginning of Section 3.5, for the sake of simplicity we initially assumed that only one expert dies each night. First, we discuss how to handle a night with more than one death. Afterwards, we explain how to extend/modify HPU and HPK to implement the Follow The Leader (FTL) strategy. We then introduce a new algorithm which simulates FTL efficiently and maintains L_t^* as well, where L_t^* is the cumulative loss of the best permutation expert through the end of round t. Finally, using L_t^*, we explain how to simulate AdaHedge and FlipFlop (de Rooij et al. (2014)) by slightly extending HPU and HPK.

More than one expert dying in a night. We handle nights with more than one death as follows. We have one of the experts die on that night, and, for each expert j among the other experts that should have died that night, we create a "dummy round", give all alive experts (including expert j) a loss of zero, keep the learning rate the same as the previous round, and have expert j die at the end of the dummy round (which hence becomes a "dummy night"). Even though the number of rounds increases with this trick, it is easy to see that the regret is unchanged since in dummy rounds all experts have the same loss (and also the learning rate after the sequence of dummy rounds is the same as what it would have been had there been no dummy rounds). Moreover, since now one expert dies on each night (some of which may be

dummy nights), we may use Theorems 9 or 10 to conclude that our algorithm correctly distributes any dying experts' weights among the alive experts.

Beyond adaptivity to m. Consider the case of unknown order and let the number of nights m be unknown. As promised, we show that we can improve on the simple Resetting-Hedge strategy.

Theorem 11 *Consider running Hedge on top of K copies of HPU where, for $r \in \{0, 1, \ldots, K-1\}$, we set $\varepsilon_r = \prod_{l=0}^{r-1} \frac{1}{K-l}$ and the r^{th} copy of HPU uses learning rate $\eta_t^{\varepsilon_r} := \sqrt{8\log(1/\varepsilon_r)/t}$. Let π^* be a best permutation expert in hindsight and suppose that the sequence $(\sigma^1(\pi^*), \ldots, \sigma^T(\pi^*))$ changes experts at most l times. Then the regret of this algorithm is $\mathcal{O}(\sqrt{T(l+1)\log K})$.*

Proof (Proof of Theorem 11) The idea is to use a simple counting argument. Let π be a best permutation expert (it typically will not be unique). For the dying sequence that actually occurs, we will lower bound how many other permutations have the same behavior as this one (we call these behavioral copies). First, observe that if there are m nights, then each permutation expert can only change the actual expert it uses for prediction at most m times (for a total of $m+1$ different experts). Suppose that, over the course of the game, π predicts as $e_{i_1}, e_{i_2}, \ldots, e_{i_l}$ where $l \leq m+1$. Now, consider those permutations that actually *begin* as $e_{i_1}, e_{i_2}, \ldots, e_{i_l}$. As the first l positions are fixed, there are $(K-l)!$ such permutations and hence at least $(K-l)!$ behavioral copies of π. Hence, if π is the best expert, then we can compete with it using an ε-quantile bound with $\varepsilon = \frac{(K-l)!}{K!} = \prod_{r=0}^{l-1} \frac{1}{K-r} = \varepsilon_l$. Although we do not know the best choice of ε, as we run Hedge on top of K copies of Hedge-Perm-Unknown and as one of the copies will use learning rate ε_l, we can compete with this optimally tuned copy with additional regret overhead of $\sqrt{T\log K}$. Moreover, a basic quantile bound exercise shows that the regret of the optimally tuned copy will be $\mathcal{O}(\sqrt{T(l+1)\log K})$, where $l \leq m$. ∎

Note that this theorem does better than adapt to m, as with m nights we always have $l \leq m$ but l can in fact be much smaller than m in practice. Hence, Theorem 11 recovers and can improve upon the regret of Resetting-Hedge and, moreover, wasteful resetting is avoided. Also, while the computation increases by a factor of K, it is easy to see that one can instead use an exponentially spaced grid of size $\log_2(K)$ to achieve regret of the same order.

Follow the Leader. FTL might be the most natural algorithm proposed in online learning. In round t the algorithm plays the expert with the lowest cumulative loss up to round t, L^*_{t-1}. By setting $\eta = \infty$ in Hedge and similarly, in HPU and HPK, we recover FTL; hence, our algorithms can simulate FTL. The motivation for FTL is that it achieves constant regret (with respect to T) when the losses are i.i.d. stochastic and there is a gap in mean loss between the best and second best (permutation) experts. Our algorithms do not maintain L^*_t, but we need L^*_t to implement AdaHedge (which we discuss in the next extension). Here, we propose a simple algorithm to perform FTL on the set of orderings. The algorithm works as follows:

1. Perform as FTL on alive initial experts and keep track of their cumulative losses $(L^t_1, L^t_2, \ldots, L^t_K)$, while ignoring the dead experts;

2. If expert j dies in round t', then for every alive expert i where $L^{t'}_i > L^{t'}_j$ do: $L^{t'}_i := L^{t'}_j$.

This not only performs the same as FTL but also explicitly keeps track of L^*_t. We will use this implementation to simulate AdaHedge.

AdaHedge. The following change to the learning rate in HPU/HPK recovers AdaHedge. Let $\hat{L}_t = \sum_{r=1}^t \hat{\ell}_r$. For round t, AdaHedge on N experts sets the learning rate as $\eta_t = (\ln N)/\Delta_{t-1}$ and $\Delta_t = \hat{L}_t - M_t$ where $M_t = \sum_{r=1}^t m_r$ and $m_r = -\frac{1}{\eta_r} \ln(\boldsymbol{w}_r \cdot e^{-\eta_r \boldsymbol{\ell}_r})$; here, m_r can easily be computed using the weights from HPU/HPK. As we have the loss of the algorithm at each round, we can calculate M_t. Also, using the implementation of FTL describe above, we can maintain L^*_t. Finally, we can compute Δ_t and the regret of HPU/HPK.

FlipFlop. By combining AdaHedge and FTL, de Rooij et al. (2014) proposes FlipFlop which can do as well as either of AdaHedge (minimax guarantees and more) or FTL (for the stochastic i.i.d. case). We can adapt HPK and HPU to FlipFlop by implementing AdaHedge and FTL as described above and switching between the two based on Δ^{ah}_t and Δ^{ftl}_t, where Δ^{ftl}_t is defined similar to Δ^{ah}_t but the learning rate associated with m_t for FTL is $\eta^{\text{ftl}} = \infty$ while for AdaHedge it is $\eta^{\text{ah}}_t = \frac{\ln K}{\Delta_{t-1}}$.

Corollary 1 *By combining FTL and AdaHedge as described above, HPU and HPK simulate FlipFlop over set of experts A (where $A = \Pi$ for HPU and $A = \mathcal{E}$ for HPK)*

and achieve regret

$$R_A(1,T) < \min\left\{ C_0 R_A^{\text{ftl}}(1,T) + C_1, C_2\sqrt{\frac{L_T^*(T-L_T^*)}{T}\ln(|A|)} + C_3\ln(|A|) \right\},$$

where C_0, C_1, C_2, C_3 are constants.

The interest in FlipFlop is that in the real-world we may not know if losses are stochastic or adversarial. This motivates one to use an algorithms that detect and adapt to easier situations.

3.6 Conclusion

In this chapter, we introduced the dying experts setting. We presented matching upper and lower bounds on the ranking regret for both the cases of known and unknown order of dying. In the case of known order, we saw that the reduction in the number of effective orderings allows our bounds to be reduced by a $\sqrt{\log K}$ factor. While it appears to be computationally hard to obtain sublinear regret in the general sleeping experts problem, in the restricted dying experts setting we provided efficient algorithms with optimal regret bounds for both cases. Furthermore, we proposed an efficient implementation of FTL for dying experts which, combined with efficiently maintaining mix losses, enabled us to extend our algorithms to simulate AdaHedge and FlipFlop. It would be interesting to see if the notion of effective experts can be extended to other settings such as multi-armed bandits. Furthermore, it might be interesting to study the problem in regimes in between known and unknown order.

Chapter 4

Online Fairness

In this chapter and the next, we study two different problems concerned with algorithmic fairness. We give an introduction to the general class of fair machine learning problems, followed by Section 4.2, where we introduce our specific setup for this chapter.

4.1 Introduction

As algorithms and machine learning methods are being employed more in helping with making decisions, a particularly important issue is provoking debates on the use of these methods in certain contexts. That is, studies and reports have repeatedly shown that autonomous decisions are "unfair" to certain groups (Barocas and Selbst (2016); Munoz et al. (2016)). Research on this topic is witnessing a rapid evolution among the machine learning community.

Some studies argue that the data used to train our methods might be the reason for discriminatory outcomes and they try to remove the underlying bias in data by pre-processing techniques (see Kamiran and Calders (2012)). On the other hand, some adopt a post-processing approach to relabel a black box algorithm's outcomes as it is done in the seminal work of Hardt et al. (2016). Our methods in Chapter 4 and 5 however, fall into the category of in-processing approaches. As the name suggests, we attempt to provide algorithms that enforce fairness during the learning process. These algorithms guarantee fairness with respect to a specific notion that is usually motivated by a higher-level philosophical argument. One prevalent notion in this regard is *equality of opportunity* (Hardt et al. (2016)), itself a relaxation of equalized

odds. For equalized odds, we require the predictor to make predictions $\hat{y}$ independent of the *protected variable* A, conditional on the true outcome y. In the context of fairness, a protected variable is often associated with race, gender, or religion. The idea behind equality of opportunity is that one does not have to equalize the odds of every outcome y, but only those that are considered "advantaged". In the context of binary prediction, these outcomes can be giving a student admission or giving an inmate parole.

For the problem in this chapter, we are specifically interested in devising a non-discriminatory algorithm for adversarial online learning. Our approach involves using Hedge algorithm (Algorithm 1). That is, we are interested in extending Hedge to obtain a non-discriminatory algorithm and, at the same time, performing as well as the best expert (in terms of achieving regret sublinear in T). In particular, we attempt to extend the analysis of Blum et al. (2018)[Theorem 3] to obtain an *any-time* fairness guarantee. In Blum et al. (2018)[Theorem 2], it is shown that achieving this goal with respect to equalized odds notion is impossible, even if Learner has access to experts that are non-discriminatory in isolation. Alternatively, they obtain a fairness guarantee for notion of *equalized error rates* (to be introduced in Section 4.2). To explain our goal in detail, let us introduce our notation and setup for the rest of this chapter. Although definitions are introduced throughout, Table 4.1 includes some of our notation for quick reference.

4.2 Problem Setup

Similar to PEA protocol in Figure 2.1, for each round t, losses of K experts are chosen adversarially in $[0, 1]$. In addition to the standard Hedge game, an example at round t belongs to a group $g_t \in \mathcal{G}$ and the group identity is accessible for Learner during the game. This setting is often referred to as *group-aware*, contrasting the group-unaware setting where algorithm does not know which group the example belongs to when making predictions; the group-unaware setting is not going to be discussed in this work. Adopting the notion of equalized error rate that is used in Blum et al. (2018), but with a small but important modification, we require the algorithm to maintain a small gap between error rates over groups throughout the game. That is, if we denote by $\widehat{\mathcal{D}}_t$ the discrepancy between normalized cumulative losses of the algorithm on group

$\ell_{i,t}$	Instantaneous loss of expert i at round t
$L_{i,t} = \sum_{s=1}^{t} \ell_{i,t}$	Cumulative loss of expert i up until round t
$L_t^* = \min_{i \in [K]} L_{i,t}$	Cumulative loss of the best expert until round t
$\lvert g_t \rvert$	Number of examples with group identity g up until round t
T	Total number of rounds at the end of the game
$w_{i,t}$	Weight placed on expert i on the beginning of round t
$\hat{\ell}_t^{(\mathcal{A})}$	Instantaneous loss of algorithm $\mathcal{A}$ at round t
$\widehat{L}_t^{(\mathcal{A})} = \sum_{s=1}^{t} \hat{\ell}_s^{(\mathcal{A})}$	Cumulative loss of algorithm $\mathcal{A}$ up until round t
$m_t^{(\mathcal{A})}$	Instantaneous mix loss of algorithm $\mathcal{A}$ at round t
$M_t^{(\mathcal{A})}$	Cumulative mix loss of algorithm $\mathcal{A}$ up until round t
$\widehat{\mathcal{D}}_t$	Discrepancy between average loss on group A and group B for Learner up until round t
$\mathcal{D}_t^*$	Discrepancy between average loss on group A and group B for the best expert up until round t

$L_{i,t}(g)$ is the cumulative loss of expert i up until round t over the rounds relevant to group g, i.e., considering only the rounds where the example belongs to group g. Subscripts i and t are relevant to "experts" and "time", respectively.

Table 4.1: Notation Summary

A and group B up until round t,

$$\widehat{\mathcal{D}}_t = \left| \frac{\widehat{L}_t(A)}{A_t} - \frac{\widehat{L}_t(B)}{B_t} \right|,$$

we require $\widehat{\mathcal{D}}_t$ to approach zero as $t \to \infty$. In particular, we are interested in $\widehat{\mathcal{D}}_t \leq c\theta_t^{-\rho}$ for some $c \geq 0$ and $0 < \rho \leq 1$ and for $t \in [T]$ where $\theta_t := \min\{|A_t|, |B_t|\}$, $|g_t|$ is the number of rounds with examples of group g by the end of round t and $\widehat{L}_t(g)$ is the cumulative loss relevant to examples from group g up until round t. In other words, we want the discrepancy between normalized losses over groups to be sublinear in the length of the under-represented group. Formally, the fairness rule we require the algorithm to satisfy is:

$$\forall t \in [T], \quad \exists c \geq 0, \quad \exists 0 < \rho \leq 1, \text{ s.t. } \widehat{\mathcal{D}}_t := \left| \frac{\widehat{L}_t(A)}{A_t} - \frac{\widehat{L}_t(B)}{B_t} \right| \leq c\theta_t^{-\rho}. \tag{4.1}$$

As one might already have noticed, this notion reflects the idea of fairness at every round as we need it to hold for every $t \in [T]$. In a very close approach, the proof in Blum et al. (2018)[Theorem 3] can be used to guarantee (4.1) but only for $\widehat{\mathcal{D}}_T$. This

means Learner guarantees fairness but only at the very end of the game. However, this might not be reasonable in real-life scenarios since Learner can be very discriminating in a certain period and start catching up on it enough rounds before the end of the game. This motivates the question of whether the approach in Blum et al. (2018)[Theorem 3] can be extended to provide an any-time guarantee. As we will show, to achieve this, our main objective will be to obtain a lower bound for the cumulative loss of Hedge with a time-varying learning rate (Section 2.3).

Before proceeding further, let us re-state Theorem 3 of Blum et al. (2018) using our notation. It is stated for two groups only, but the result extends to more than two groups.

Theorem 12 (Theorem 3 Blum et al. (2018)) *Running separate instances of Hedge algorithm for each group $g \in \{A, B\}$, there exist constants $c \geq 0$ and $0 < \rho \leq 1$ such that (4.1) is guaranteed for $t = T$ as well as regret sublinear in T.*

We use the same proof strategy as in the original work, with two modifications. Denote by $\widehat{L}_T^{\eta_T}$ the cumulative loss of Hedge that uses fixed learning rate of η_T and by L_T^* the cumulative loss of the best expert. The first difference is that we relax the assumption in Blum et al. (2018) about experts being fair in isolation. In Blum et al. (2018), they assume the following holds for all the experts $j \in [K]$:

$$\left| \frac{L_{j,T}(A)}{|A_T|} - \frac{L_{j,T}(B)}{|B_T|} \right| = 0.$$

While instead, we assume the following:

$$\forall j \in [K], \quad \exists c_j \geq 0, \quad \exists 0 < \rho_j \leq 1, \text{s.t.} \quad \left| \frac{L_{j,T}(A)}{|A_T|} - \frac{L_{j,T}(B)}{|B_T|} \right| \leq c_j (\theta_T)^{-\rho_j}. \tag{4.2}$$

Note that this only holds for the very end of the game (round T). The second difference is that we will improve the inequality in (4.4) below to $\widehat{L}_T^{(\eta_T)} \geq L_T^*$ later in Claim 13.

Proof Sketch

The proof for Theorem 12 relies on two properties of Hedge algorithm with fixed learning rate η_T: :

$$\widehat{L}_T^{(\eta_T)} \leq (1 + \eta_T) L_T^* + \frac{\ln(K)}{\eta_T} \tag{4.3}$$

$$\widehat{L}_T^{(\eta_T)} \geq (1 - 4\eta_T) L_T^* \tag{4.4}$$

With this strategy, these are the two key ingredients needed. The rest is plugging (4.3) and (4.4) both in the fairness rule (4.1) using $t = T$ since we are only obtaining the guarantee for the very end of the game. We have:

$$\begin{aligned}
\widehat{\mathcal{D}}_T &= \left| \frac{\widehat{L}_T^{(\eta_T)}(A)}{|A_T|} - \frac{\widehat{L}_T^{(\eta_T)}(B)}{|B_T|} \right| \\
&\leq \left| \frac{(1+\eta_T)L_T^*(A) + \ln(K)/\eta_T}{|A_T|} - \frac{L_t^*(B) - 4\eta_T L_T^*}{|B_T|} \right| \\
&= \left| \frac{L_T^*(A)}{|A_T|} - \frac{L_T^*(B)}{|B_T|} + \frac{\eta_T L_T^*(A)}{|A_T|} + \frac{4\eta_T L_T^*(B)}{|B_T|} + \frac{\ln(K)}{\eta_T |A_T|} \right| \\
&\leq \left| \mathcal{D}_T^* + 5\eta_T + \frac{\ln(K)}{\eta_T |A_T|} \right|. \qquad (4.5)
\end{aligned}$$

where $\mathcal{D}_T^*$ the discrepancy between average loss on group A and B for the best expert. Choosing $\eta_T = \sqrt{\frac{\ln K}{5|A_T|}}$ yields:

$$\widehat{D}_T \leq c_j(\theta_T)^{-\rho_j} + 6\sqrt{\frac{\ln K}{5|A_T|}}$$

where j is the index of best expert. ∎

As mentioned before, the result can be improved (by a constant multiplicative factor) using our improvement of (4.4) in Claim (13). With these explanations, we will try to extend this strategy to guarantee fairness for every round $t \in [T]$. This leads to an interesting question about Hedge algorithm with time-varying learning rate, one that has not been answered before.

4.3 Any-time fairness guarantee

As mentioned, to obtain an any-time fairness guarantee, we follow the same approach as in the proof for Theorem 12 except we use exponential weights with time-varying learning rate $\eta_t := \sqrt{8\log(K)/t}$. Since $\eta_1 > \eta_2 > \cdots > \eta_T$, we refer to this algorithm as decreasing Hedge (`dec`) in the following (Mourtada (2019)). So as an example, the cumulative loss of this algorithm will be denoted by $\widehat{L}_T^{(\texttt{dec})}$.

In order to follow the same strategy as in the proof of Theorem 12, we need to establish properties similar to (4.3) and (4.4) for `dec` algorithm for every t. But before that, let us find out what bounds we are satisfied with. Note that we already have an

upper bound from Theorem 2, resulting in (4.6) immediately. We construct the form of the lower bound we want with a variable Λ; then we continue with the fairness proof and find out the minimal value for Λ we are satisfied with. So we have:

$$\begin{aligned} \widehat{L}_t^{(\texttt{dec})} &\leq L_t^* + \sqrt{2t \ln K} + \sqrt{\frac{\ln K}{8}} \\ \widehat{L}_t^{(\texttt{dec})} &\geq L_t^* - \Lambda. \end{aligned} \tag{4.6}$$

To show the fairness guarantee, we follow steps as:

$$\begin{aligned} \widehat{\mathcal{D}}_t &\leq \frac{L_t^*(A) + \sqrt{2t \ln K} + \sqrt{\ln(K)/8}}{|A_t|} - \frac{L_t^*(B) - \Lambda}{|B_t|} \\ &= \frac{L_t^*(A)}{|A_t|} - \frac{L_t^*(B)}{|B_t|} + \frac{\sqrt{2t \ln K} + \sqrt{\ln(K)/8}}{|A_t|} + \frac{\Lambda}{|B_t|} \\ &= \mathcal{D}_t^* + \frac{(\sqrt{2t} + 1/8)\sqrt{\ln K}}{|A_t|} + \frac{\Lambda}{|B_t|} \end{aligned} \tag{4.7}$$

Assuming that we have enough examples of each group, i.e., $|g_t| = \Omega(t)$ for $g \in \mathcal{G}$, one can easily verity that, in order to satisfy $\widehat{\mathcal{D}}_t = o(t)$, it suffices Λ to be $o(t)$ as well. So our objective is reduced to obtain a lower bound for the cumulative loss of `dec` algorithm up until round t that is sublinear in t. Note that we want to obtain an any-time guarantee, meaning, the lower bound on $\widehat{L}_t^{\texttt{dec}}$ should hold for all $t \in [T]$.

4.4 Lower bound on cumulative loss of Hedge

As described earlier, our goal is to answer the following question:

Question 1 *For Hedge using time-varying learning rate $\eta_t := \sqrt{8 \log(K)/t}$ and using and weight updates as*

$$w_{j,t} = e^{-\eta_t L_{j,t-1}}$$

can one construct a lower bound as

$$\widehat{L}_t^{(\texttt{dec})} \geq L_t^* - \Lambda \tag{4.8}$$

where Λ is sublinear in t?

Recall the probability of playing each expert j by the dec algorithm at round t is:

$$\hat{p}_{j,t} = \frac{w_{j,t}}{\sum_{j\in[K]} w_{j,t}}$$

and the classical notion of regret for t rounds is defined as

$$\mathcal{R}_t = \underbrace{\sum_{j=1}^{K}\sum_{s=1}^{t}\hat{p}_{j,s}\ell_{j,s}}_{\widehat{L}_t} - \underbrace{\min_{j\in[K]}\sum_{s=1}^{t}\ell_{j,s}}_{L_t^*}. \tag{4.9}$$

We will use a method called mix loss approximation (De Rooij et al. (2014)) to achieve such bound (the final bound will appear in (4.18)). Let $\hat{\ell}_t$ denote the instantaneous loss suffered by the algorithm at (the end of) round t, i.e. $\hat{\ell}_t = \sum_j \hat{p}_{j,t}\ell_{j,t}$; we often refer to this as Hedge loss. Define instantaneous mix loss at round t as

$$m_t = -\frac{1}{\eta_t}\ln\left(\mathsf{E}_{j\sim p_t}\left[e^{-\eta_t\ell_{j,t}}\right]\right). \tag{4.10}$$

Observe that $\hat{\ell}_t = m_t + \delta_t$ where δ_t is referred to as *mixability gap* (De Rooij et al. (2014)). Define cumulative mix loss of dec algorithm up until round t as $M_t^{(\mathsf{dec})} = \sum_{s=1}^{t} m_s$ and cumulative mixability gap $\Delta_t = \sum_{s=1}^{t}\delta_s$. Therefore we have:

$$\widehat{L}_t^{(\mathsf{dec})} = M_t^{(\mathsf{dec})} + \Delta_t$$

From Property #1 in Lemma 1 of De Rooij et al. (2014), we have Δ_t is non-negative (and non-decreasing in t). Thus, we have $\widehat{L}_t^{(\mathsf{dec})} \geq M_t^{(\mathsf{dec})}$. Therefore, to obtain (4.8) it suffices to lower bound cumulative mix loss ($M_t^{(\mathsf{dec})}$) instead of cumulative Hedge loss ($\widehat{L}_t^{(\mathsf{dec})}$).

Denote by Π uniform distribution and by $M_t^{(\eta)}$ the cumulative mix loss of an algorithm using *fixed* learning rate as η and uniform prior on experts, i.e.:

$$M_t^{(\eta)} = -\frac{1}{\eta}\ln\left(\mathsf{E}_{j\sim\Pi}\left[e^{-\eta L_{j,t}}\right]\right)$$

Before proceeding further, let us find out what is the equivalent lower bound we seek for dec algorithms in the case of algorithms using fixed learning rate of η.

Claim 13 *For the case of algorithms with fixed learning rate η_t we have $\widehat{L}_t^{(\eta_t)} \geq L_t^*$.*

The result in the claim above is a well-known one (De Rooij et al. (2014)) and the proof for it is deferred to the end of this section. We can re-write the cumulative mix loss of `dec` algorithm ($M_T^{(\mathtt{dec})}$) in terms of cumulative mix losses of algorithms with fixed learning rates ($M_t^{(\eta_t)}$ under uniform distribution) as it is more convenient to analyze the mix loss for an algorithm with fixed learning rate.

$$M_T^{(\mathtt{dec})} = \sum_{t=1}^{T} \left(M_t^{(\eta_t)} - M_{t-1}^{(\eta_t)} \right) \tag{4.11}$$

Let us re-visit the proof of Lemma 2 of De Rooij et al. (2014), where they obtain an *upper* bound (we need a lower bound). Note that if we manage to replace $M_{t-1}^{(\eta_t)}$ by $M_{t-1}^{(\eta_{t-1})}$, the summation below telescopes to $M_T^{(\eta_T)}$.

$$M_T^{(\mathtt{dec})} = \sum_{t=1}^{T} \left(M_t^{(\eta_t)} - M_{t-1}^{(\eta_t)} \right) \leq \sum_{t=1}^{T} \left(M_t^{(\eta_t)} - M_{t-1}^{(\eta_{t-1})} \right) = M_T^{(\eta_T)} \tag{4.12}$$

where the inequality $M_{t'}^{(\eta_t)} \leq M_{t'}^{(\eta_{t+1})}$ comes from Property #4 of Lemma 1 in De Rooij et al. (2014). To explain more, Property #4 shows that the cumulative mix loss is non-increasing in η, and we know that $\eta_t > \eta_{t+1}$. So if we obtain the opposite side of this property, we have a lower bound on $M_T^{(\mathtt{dec})}$ in terms of $M_T^{(\eta_T)}$, and using the proof of Claim 13, we can achieve (4.8). Therefore, the goal reduces to obtaining a bound on differences of cumulative mix losses of two algorithms with learning rates η_t and η_{t+1}, i.e., an upper bound on $M_{t'}^{(\eta_{t+1})} - M_{t'}^{(\eta_t)}$. The result is stated below.

Lemma 6 *For $0 \leq \eta_{t+1} < \eta_t$ and $t' = 1, 2, \ldots$ we have:*

$$M_{t'}^{(\eta_{t+1})} - M_{t'}^{(\eta_t)} \leq \frac{1}{t'} M_{t'}^{(\eta_t)}$$

To prove the above, we first establish a technical result as stated in Lemma 7. Define $h_{j,t}^{(\eta)} := \frac{1}{\eta}\left(1 - e^{-\eta L_{j,t}}\right)$. Define generalized Hellinger divergence (refer to Grünwald and Mehta (2020) for more background on this) of order η as:

$$H_t^{(\eta)} = \mathsf{E}_{j \sim \Pi}\left[h_{j,t}^{(\eta)} \right]. \tag{4.13}$$

Note the definition of Hellinger divergence as above is simplified; we originally have:

$$H_t^{(\eta)} = \mathsf{E}_{j \sim \Pi}\left[\frac{1}{\eta}\left(1 - e^{-\eta(L_{j,t} - 0)}\right) \right]$$

where 0 corresponds to the loss of some (not necessarily realizable/obtainable) predictor that always obtains cumulative loss of zero.

Lemma 7 *For $0 \leq \eta < \eta' < \bar{\eta}$, we have $H_t^{(\eta)} \leq C H_t^{(\eta')}$ where:*

$$C = \left(\frac{\bar{\eta}}{\eta} - 1\right) / \left(\frac{\bar{\eta}}{\eta'} - 1\right)$$

Proof (Lemma 7) In what follows, all expectations are taken with respect to $j \sim \Pi$. Define $S_{j,t}^{(\eta)} = h_{j,t}^{(\eta)} - h_{j,t}^{(\bar{\eta})}$. Observe that $S_{j,t}^{(\eta)}$ is non-negative. We thus have:

$$H_t^{(\eta)} = \mathsf{E}\left[h_{j,t}^{(\eta)} - h_{j,t}^{(\bar{\eta})} + h_{j,t}^{(\bar{\eta})}\right] = \mathsf{E}\left[S_{j,t}^{(\eta)}\right] + H_t^{(\bar{\eta})} \tag{4.14}$$

Now, we try to establish for some constant C the following upper bound:

$$\mathsf{E}\left[S_{j,t}^{(\eta)}\right] \leq C\,\mathsf{E}\left[S_{j,t}^{(\eta')}\right] \tag{4.15}$$

It suffices to try to find some C such that, for all $j \in [K]$,

$$\frac{1}{\eta}\left(1 - e^{-\eta L_{j,t}}\right) - \frac{1}{\bar{\eta}}\left(1 - e^{-\bar{\eta} L_{j,t}}\right) \leq \frac{C}{\eta'}\left(1 - e^{-\eta' L_{j,t}}\right) - \frac{1}{\bar{\eta}}\left(1 - e^{-\bar{\eta} L_{j,t}}\right)$$

which, defining $r = e^{-\bar{\eta} L_{j,t}}$, is equivalent to:

$$\frac{1}{\bar{\eta}}\left(\frac{1}{\eta/\bar{\eta}}\left(1 - r^{\eta/\bar{\eta}}\right) - (1 - r)\right) \leq \frac{C}{\bar{\eta}}\left(\frac{1}{\eta'/\bar{\eta}}\left(1 - r^{\eta'/\bar{\eta}}\right) - (1 - r)\right)$$

Next, define the function $g^{\bar{\eta}}(r) = \frac{1}{\bar{\eta}}(1 - r^{\bar{\eta}}) - (1 - r)$. Then our goal can be restated as establishing (for some constant C) the inequality:

$$g^{\eta/\bar{\eta}}(r) \leq C g^{\eta'/\bar{\eta}}(r).$$

For this, we can use the first part of Lemma 36 of Grünwald and Mehta (2020). Applying the result, we can take $C = \left(\frac{\bar{\eta}}{\eta} - 1\right) / \left(\frac{\bar{\eta}}{\eta'} - 1\right)$. We thus have established (4.15). In particular, we have $C \geq 1$ (which is as it should be, as the reverse inequality

holds when C is not present at all). Continuing from (4.14), we have:

$$\begin{aligned} H_t^{(\eta)} &\leq C\,\mathsf{E}\left[S_{j,t}^{(\eta')}\right] + H_t^{(\bar{\eta})} \\ &= C\left(H_t^{(\eta')} - H_t^{(\bar{\eta})}\right) + H_t^{(\bar{\eta})} \\ &= CH_t^{(\eta')} - (C-1)H_t^{(\bar{\eta})} \end{aligned}$$

Now, we use the fact that $H_t^{(\bar{\eta})} \geq 0$. To see this, observe that:

$$H_t^{(\bar{\eta})} = \mathsf{E}\left[h_{j,t}^{(\bar{\eta})}\right] = \frac{1}{\bar{\eta}}\left(1 - \mathsf{E}\left[e^{-\bar{\eta}L_{j,t}}\right]\right),$$

where $\mathsf{E}[e^{-\bar{\eta}L_{t,j}}] \leq 1$ as the losses are non-negative. Combined with the fact that $C \geq 1$, it follows that

$$H_t^{(\eta)} \leq CH_t^{(\eta')}$$

■

Now using Lemma 7 we prove Lemma 6. Note that using Lemma 7 for $0 \leq \eta_{t+1} < \eta_t < \bar{\eta}$, we can take $\bar{\eta} \to \infty$ obtaining $C = \frac{\eta_t}{\eta_{t+1}}$.

Proof (Lemma 6) Define generalized Rényi divergence:

$$\mathsf{D}_t^{(\eta)} = -\frac{1}{\eta}\log\left(\mathsf{E}_{j\sim\Pi}\left[e^{-\eta L_{j,t}}\right]\right)$$

This is equivalent to the definition of cumulative mix loss of an algorithm using fixed learning rate of η over t rounds. Note that again, similar to the way we defined Hellinger divergence, we are using a simplified definition of Rényi divergence[1]. Observe that Rényi divergence is a one-to-one transformation of Hellinger divergence:

$$H^{(\eta_t)} = \frac{1}{\eta}\left(1 - e^{-\eta\mathsf{D}^{(\eta_t)}}\right) \tag{4.16}$$

Transforming the bound in Lemma 7 using (4.16), we have:

$$\frac{1}{\eta_{t+1}}\left(1 - e^{-\eta_{t+1}\mathsf{D}_{t'}^{(\eta_{t+1})}}\right) \leq C\frac{1}{\eta_t}\left(1 - e^{-\eta_t\mathsf{D}_{t'}^{(\eta_t)}}\right)$$

As $M_t^{(\eta)} = \mathsf{D}_t^{(\eta)}$, substituting $C = \eta_t/\eta_{t+1}$, and using inequality $\frac{\sqrt{t+1}}{\sqrt{t}} \leq 1 + 1/t$, above

[1] Originally it is defined as: $\mathsf{D}_t(\eta) := -\frac{1}{\eta}\log\left(\mathsf{E}_{j\sim\Pi}\left[e^{-\eta L_{j,t}-0}\right]\right)$.

inequality becomes:

$$M_t^{(\eta_{t+1})} \leq \frac{\eta_t}{\eta_{t+1}} M_t^{(\eta_t)} \leq \frac{\sqrt{t+1}}{\sqrt{t}} M_t^{(\eta_t)} \leq \left(1 + \frac{1}{t}\right) M_t^{(\eta_t)}$$

Finally, we have the bound:

$$M_{t'}^{(\eta_t)} - M_{t'}^{(\eta_{t+1})} \geq -\frac{1}{t'} M_{t'}^{(\eta_t)} \tag{4.17}$$

■

Now we show a proof for Claim 13 for the sake of completeness.

PROOF (CLAIM 13) By definition, we have:

$$\widehat{L}_t^{(\eta_t)} = \left(\widehat{L}_t^{(\eta_t)} - M_t^{(\eta_t)}\right) + \left(M_t^{(\eta_t)} - L_t^*\right) + L_t^*$$

The term $\widehat{L}_t^{(\eta_t)} - M_t^{(\eta_t)}$ is cumulative mixability gap Δ_t, which is always non-negative. This, combined with $M_t^{(\eta_t)} \geq L_t^*$, proves the claim. It is easy to verify the latter inequality:

$$\begin{aligned}
M_t^{(\eta_t)} &= \sum_{s=1}^{t} -\frac{1}{\eta_t} \log\left(\mathsf{E}_{j \sim p_t^{(\eta_t)}}[e^{-\eta_t \ell_{j,s}}]\right) \\
&= -\frac{1}{\eta_t} \sum_{s=1}^{t} \log\left(\frac{\sum_{j=1}^{K} \pi(j) e^{-\eta_t L_{j,s}}}{\sum_{j=1}^{K} \pi(j) e^{-\eta_t L_{j,s-1}}}\right) \\
&= -\frac{1}{\eta_t} \log \sum_{j=1}^{K} \pi(j) e^{-\eta_t L_{j,t}} \qquad \text{(Telescoping)} \\
&\geq -\frac{1}{\eta_t} \log\left(\frac{1}{K} \sum_{j=1}^{K} \pi(j) e^{-\eta_t L_t^*}\right) \\
&= L_t^*
\end{aligned}$$

■

Finally, applying Lemma 6 on (4.11) for t' rounds, we get the following bound:

$$\begin{aligned} M_{t'}^{(\mathsf{dec})} &= \sum_{t=1}^{t'} \left(M_t^{(\eta_t)} - M_{t-1}^{(\eta_t)} \right) \\ &\geq \sum_{t=1}^{t'} \left(M_t^{(\eta_t)} - \left(1 + \frac{1}{t}\right) M_{t-1}^{(\eta_{t-1})} \right) \\ &= \sum_{t=1}^{t'} \left(M_t^{(\eta_t)} - M_{t-1}^{(\eta_{t-1})} \right) - \sum_{t=1}^{t'} \left(\frac{1}{t} M_{t-1}^{(\eta_{t-1})} \right) \\ &= M_{t'}^{(\eta_{t'})} - \sum_{t=1}^{t'} \left(\frac{1}{t} M_{t-1}^{(\eta_{t-1})} \right) \end{aligned} \tag{4.18}$$

Let us explain how this bound is not quite useful for us. In (4.18), the term $\sum_{t=1}^{t'} \frac{1}{t} M_{t-1}^{(\eta_{t-1})}$ can grow linear in t' when $M_t = t$. This happens when Learner places all the weights on one expert and that expert suffers loss 1 at every round. Therefore, we have:

$$M_{t'}^{(\mathsf{dec})} \geq M_{t'}^{(\eta_{t'})} - ct'$$

for some constant $c > 0$. Unfortunately, this bound is looser than the minimal one needed for our fairness guarantee to work. Recall that we wanted a bound as stated in Question 1 with $\Lambda = o(t')$, while using $L_{t'}^{(\mathsf{dec})} \geq M_{t'}^{(\mathsf{dec})}$, we can only obtain this with $\Lambda = \Omega(t')$. This leaves our Question 1 open. Note that our main motivation was to provide an any-time guarantee for fairness in our setup. This means that there might be other approaches that do not require obtaining such a lower bound in the first place.

Chapter 5

Envy-Free Fairness

5.1 Introduction

Consider the classical, supervised learning setting where m examples with their corresponding labels $(X_1, Y_1), \ldots, (X_m, Y_m)$ belong to sample space $\mathcal{Z}$ and are drawn i.i.d. from an unknown probability distribution. The learning algorithm has access to a function class $\mathcal{F}$ and each example belongs to a group $g \in \mathcal{G}$. For simplicity, we often assume $\mathcal{G} = \{A, B\}$. Imagine prediction on one group is easier than the other, meaning Learner can find a hypothesis that has a low expected loss on one group but no such low-risk hypothesis exists for the other group. In this scenario, constraining Learner to satisfy certain notions of fairness might hurt accuracy, e.g., to equalize false negative rates, Learner might try to add noise to predictions on the easier group to increase false negative rate on that group (Pleiss et al. (2017)).

We devise a new notion of group fairness for supervised learning based on the concept of envy. In simple words, we restrict the algorithm from offering gain to one group at the expense of another group. This should result in a state where no group envies the other. In a recent work, Samadi et al. (2018) adopt a similar notion of fairness for a dimensionality reduction method (principal component analysis). In their work, they minimize additional average reconstruction error for each group. A similar notion have been discussed in (Williamson and Menon (2019)) where they introduce *fairness risk measures*. We explain our notion of fairness formally in the next section.

5.2 Problem Setup

In our setup, we assume there are $|\mathcal{G}|$ distributions ($|\mathcal{G}|= 2$, for now), each one corresponding to a group. We require the algorithm to output a hypothesis $\hat{f} \in \mathcal{F}$ that minimizes the maximum *excess risk* among all groups where $\mathcal{F}$ is a function class (or hypothesis space). Formally, Learner's objective is to solve

$$\min_{\hat{f}\in\mathcal{F}} \max_{g\in\mathcal{G}} \left\{ R_g(\hat{f}) - \min_{f\in\mathcal{F}} R_g(f) \right\},$$

where $R_g(f)$ is the risk of function f on examples from group g drawn from some distribution P_g. Formally, given a probability distribution P_g, for a loss function $\ell(\hat{y}, y)$, we define

$$R_g(f) = \mathsf{E}_{(X,Y)\sim P_g} \left[\ell \left(f(X), Y \right) \right]. \tag{5.1}$$

Note that we assume examples are i.i.d. in their respective groups, e.g., examples belonging to group g are drawn i.i.d. from distribution P_g, so the risk function R_g is subscripted with g. Using a simpler notation, the objective for two groups $\{A, B\}$ is as follows:

$$\min_{\hat{f}\in\mathcal{F}} \max_{g\in\{A,B\}} \left\{ R_g(\hat{f}) - R_g^* \right\}. \tag{5.2}$$

For the objective (5.2), some interesting questions need to be answered. The main question that we will answer in this work is given R_g^* for $g \in \mathcal{G}$, how can we solve for the objective in (5.2). We follow the analysis of Nemirovski et al. (2009)[Section 3] for stochastic saddle point problems. We will discuss further challenges and possible solutions at the end of this chapter.

5.2.1 Warm-up Regression Example

For simplicity, we start with regression and as already mentioned, we assume examples belong to two groups A and B. Learner have access to $n(A)$ and $n(B)$ examples drawn from unknown probability distributions P_A and P_B, respectively. We assume the risk function R_g for group g is L_g-smooth and σ_g-strongly convex over $\mathcal{Z}$. It is easy to see that the excess risk functions for each group are also strongly convex and smooth with same parameters.

Definition 1 (Strong Convexity) *A function $f : S \to \mathbb{R}$ is σ-strongly convex over S with respect to a norm $\|\cdot\|$ if there exists a constant $\sigma > 0$ such that for any $\boldsymbol{\theta}, \boldsymbol{\theta'} \in S$,*

we have:

$$f(\boldsymbol{\theta}) \geq f(\boldsymbol{\theta'}) + \langle \nabla f(\boldsymbol{\theta'}), \boldsymbol{\theta} - \boldsymbol{\theta'} \rangle + \frac{\sigma}{2} \|\boldsymbol{\theta} - \boldsymbol{\theta'}\|^2$$

Definition 2 (Smoothness) *A function $f : S \to \mathbb{R}$ is L-smooth if for any $\boldsymbol{\theta}, \boldsymbol{\theta'} \in S$, we have:*

$$\|\nabla f(\boldsymbol{\theta}) - \nabla f(\boldsymbol{\theta'})\|_2 \leq L \|\boldsymbol{\theta} - \boldsymbol{\theta'}\|_2$$

In a regression task, Learner outputs a vector $\hat{\boldsymbol{\theta}} \in \mathbb{R}^d$ and the predictions are of the form $\langle \hat{\boldsymbol{\theta}}, \boldsymbol{x} \rangle$, which is a mapping from sample space to $\mathbb{R}$, and the goal is to minimize the risk with respect to P. Now for groups $g \in \mathcal{G}$, denote the true risk minimizers by $\boldsymbol{\theta}^*_g$. Also, let us denote by $C(\boldsymbol{\theta})$ the cost of $\boldsymbol{\theta}$ defined as:

$$C(\boldsymbol{\theta}) = \max_{g \in \mathcal{G}} \{R_g(\boldsymbol{\theta}) - R_g(\boldsymbol{\theta}^*_g)\}. \tag{5.3}$$

Denote by $\boldsymbol{\theta}^*$ (without a subscript) the parameter vector that optimizes the function in (5.3), namely:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} C(\boldsymbol{\theta})$$

As a warm-up exercise, first we show that if the best predictors for two groups (which can be seen as solutions for two problems) are far from each other, which now translates to $\|\boldsymbol{\theta}^*_A - \boldsymbol{\theta}^*_B\|^2$ being large, then $\boldsymbol{\theta}^*$ is not too close to any of $\boldsymbol{\theta}^*_g$. As a result, using strong convexity, the excess risk for each group will be large. As mentioned earlier, assume the risk functions R_g are σ_g-strongly convex and L_g-smooth for all $g \in \{A, B\}$. This implies strong convexity and smoothness for excess risk functions with the same parameters. Define $D_g = \left\|\boldsymbol{\theta}^* - \boldsymbol{\theta}^*_g\right\|$ and $D = \|\boldsymbol{\theta}^*_A - \boldsymbol{\theta}^*_B\|$. From strong convexity of the excess risk, for group A we have:

$$D_A^2 \leq \frac{2}{\sigma_A} \left(\underbrace{(R_A(\boldsymbol{\theta}^*) - R_A(\boldsymbol{\theta}^*_A))}_{\text{excess risk of } \boldsymbol{\theta}^*} - \underbrace{(R_A(\boldsymbol{\theta}^*_A) - R_A(\boldsymbol{\theta}^*_A))}_{\text{excess risk of } \boldsymbol{\theta}^*_A} \right) \tag{5.4}$$

$$= \frac{2}{\sigma_A}((R_A(\boldsymbol{\theta}^*) - R_A(\boldsymbol{\theta}^*_A)). \tag{5.5}$$

Also assume:

$$\underbrace{R_A(\boldsymbol{\theta}^*) - R_A(\boldsymbol{\theta}^*_A)}_{\text{excess risk of } \boldsymbol{\theta}^* \text{over group} A} = \underbrace{R_B(\boldsymbol{\theta}^*) - R_B(\boldsymbol{\theta}^*_B)}_{\text{excess risk of } \boldsymbol{\theta}^* \text{over group} B}. \tag{5.6}$$

From smoothness of R_B we have:

$$D_B \geq \frac{1}{L_B} \left\| \nabla (R_B(\boldsymbol{\theta}^*) - R_B(\boldsymbol{\theta}_B^*)) \right\|. \tag{5.7}$$

From Lemma 3.4 Bubeck et al. (2015), and assuming the gradient at the minimizer $(\boldsymbol{\theta}_B^*)$ is equal to zero, we have:

$$D_B^2 \geq \frac{2}{L_B} (R_B(\boldsymbol{\theta}^*) - R_B(\boldsymbol{\theta}_B^*)), \tag{5.8}$$

which combined with (5.5) and (5.6), results in:

$$D_B \geq \sqrt{\frac{\sigma_A}{L_B}} D_A. \tag{5.9}$$

We can obtain a lower bound for D_A in a similar way, which will end up being:

$$D_A \geq \sqrt{\frac{\sigma_B}{L_A}} D_B. \tag{5.10}$$

Finally, using the triangle inequality as $D_A + D_B \geq D$, we obtain a lower bound for each D_g in terms of D:

$$D_A \geq \frac{\sqrt{\sigma_B}}{\sqrt{\sigma_B} + \sqrt{L_A}} D, \qquad D_B \geq \frac{\sqrt{\sigma_A}}{\sqrt{\sigma_A} + \sqrt{L_B}} D. \tag{5.11}$$

As mentioned earlier, we conclude that the excess risk for each group will be large (based on strong convexity). In the next section, we offer a solution that provably approximately solves the objective in (5.2). We use mirror descent algorithm for this problem that can be framed as stochastic saddle point approximation.

5.3 Mirror Descent Method

Before introducing the algorithm, let us review some basic definitions and results.

Definition 3 (Sub-gradients) *A vector d is a sub-gradient of a convex function f at θ' if for $\forall \theta$:*

$$f(\theta) \geq f(\theta') + \langle d, (\theta - \theta') \rangle.$$

Definition 4 (Sub-differential) *The non-empty compact convex set of all sub-gradients of function f at θ' is called the sub-differential of f at θ', denoted by $\partial f(\theta')$:*

$$\partial f(\theta') = \{d \mid \forall \theta : f(\theta) \geq f(\theta') + \langle d, (\theta - \theta') \rangle\}.$$

As a warm-up example, sub-differential of $\max\{f(\theta'), g(\theta')\}$ is calculated as below:

$$\partial \max\{f(\theta'), g(\theta')\} = \begin{cases} \nabla f(\theta') & \text{if } f(\theta') \geq g(\theta') \\ \nabla g(\theta') & \text{if } f(\theta') \leq g(\theta') \\ \underbrace{\alpha \nabla f(\theta') + (1-\alpha)\nabla g(\theta')}_{\forall 0 \leq \alpha \leq 1} & \text{if } f(\theta') = g(\theta') \end{cases}$$

which simply is every convex combination of $\nabla \arg\max\{f_1(\theta'), f_2(\theta')\}$. In the following theorem, we consider a generalization of the above example. The next result is a re-statement of Corollary 4.3.2 of Hiriart-Urruty (2001).

Theorem 14 (Hiriart-Urruty (2001)) *Define m convex functions $f_1(\theta), \ldots, f_m(\theta)$ such that $f_i : \mathbb{R}^n \to \mathbb{R}$ for $i \in [m]$. Let $f(\theta) := \max\{f_1(\theta), \ldots, f_m(\theta)\}$. Denoting by $I(\theta) := \{i : f_i(\theta) = f(\theta)\}$ the active group, for sub-differential of $f(\theta)$ we have:*

$$\partial f(\theta) = conv\{\cup \partial f_i(\theta) : i \in I(\theta)\}$$

As mentioned earlier, we assume we are given minimum achievable risks for each group, i.e., R_g^* for each $g \in \mathcal{G}$. Let us recall the objective:

$$\min_{f \in \mathcal{F}} \max_{g \in \mathcal{G}} \left\{ R_g(f) - R_g^* \right\}. \tag{5.12}$$

Denote by Δ the probability simplex defined as $\Delta = \{\lambda \in \mathbb{R}^{|\mathcal{G}|} : \sum_{g \in \mathcal{G}} \lambda_g = 1, \lambda_g \geq 0\}$. We can linearize the objective as below:

$$\min_{\theta \in \Theta} \max_{\lambda \in \Delta} \left\{ \sum_{g \in \mathcal{G}} \lambda_g \left(R_g(\theta) - R_g^* \right) \right\}. \tag{5.13}$$

We use mirror descent to optimize (5.13) similar to Section 3.2 of Nemirovski et al. (2009). Mirror descent is a general first order optimization method, since it it carries out its updates in a dual space. The following proposition states the algorithm that approximates the objective in (5.13).

Proposition 1 *The objective in (5.13) can provably be approximated using mirror descent with updates as*

$$\lambda_{j+1,g} = \frac{\lambda_{j,g} \exp\{-\gamma \ell_g(\theta_j, \xi_j^g)\}}{\sum_{g \in \mathcal{G}} \lambda_g \exp\{-\gamma \ell_g(\theta_j, \xi_j^g)\}}$$
$$\theta_{j+1,g} = \theta_g - \gamma \ell_g(\theta_j, \xi_j^g)$$

where $\xi = (\xi^1, \ldots, \xi^m)$ *is a sample vector and* $m = |\mathcal{G}|$. □

The derivation of above algorithm is essentially from Nemirovski et al. (2009). We show a complete proof and a discussion on error rate analysis in Section 5.5.

5.3.1 Problem of evaluating the maximum

Note that to implement the algorithm described above and achieve the desired error rate, one needs a sample $\xi_j = (\xi^1, \ldots, \xi^m)$ at round j, i.e., one sample from each group at each round. However, it makes more sense to think of a way to estimate the soft-max function at each round j using fewer samples. In the following, we assume an oracle outputs the maximum coordinate at each round. Later in Section 5.4, we discuss an idea that might be useful in order to obtain an algorithm without assuming the existence of such an oracle.

Oracle-based algorithm As discussed previously, our algorithm needs to evaluate a soft-max function at every step, which is expensive as it requires one sample from every group at a given round. As it will be seen in (5.32), we need to evaluate $\ell_g(\theta_j, \xi_j^g)$ for our dual updates for every g. The intuition behind the multiplicative weights updates in (5.32) is putting more weight on the group that is more probable to be the real solution of the maximum part of the objective in (5.13); this is evident in the definition of $\mathsf{G}_\lambda(\lambda, \xi)$ in (5.18) as well. We will show the algorithm in a simple case, making the assumption below:

Assumption 1 *At round* j*, Learner has access to an oracle which, for a given sample* $\xi_j = (\xi_j^1, \ldots, \xi_j^m)$ *and vector* $\theta_j \in \Theta$*, returns* $\tilde{g}_j = \arg\max_{g \in \mathcal{G}} \{\mathsf{E}_{\xi_j^g \sim P_g}[\ell_g(\theta_j, \xi_j^g)] - R_g^*\}$.

Since R_g^* in the above has no effect on $\tilde{g}_j$, we might drop it from our notation in what follows. With this assumption, the objective in (5.13) becomes a simpler objective and one only needs to perform the primal updates. In this case, to implement

the algorithm, we only need to update the gradients only with respect to $\tilde{g}_j$. We still need to show a connection between sub-gradients of $\ell_{\tilde{g}}(\theta_j, \xi_j^{\tilde{g}})$ (where $\tilde{g}$ is obtained from the oracle) can be used for our objective, which involves the expectation of this variable (the risk function). We show this in the following lemma. Recall that $R_g(f) = \mathsf{E}_{\xi \sim P_g}[\ell_g(\theta, \xi^g)]$.

Lemma 8 *Let $F(\theta) = \max_{g \in \mathcal{G}}\{R_g(\theta)\}$ and define active groups as $\mathcal{G}(\theta) = \{g \in \mathcal{G} : R_g(\theta) = F(\theta)\}$. If an oracle given θ correctly provides an element of $\mathcal{G}(\theta)$, then any element of $\partial \ell_g(\theta, \xi^g)$ is an unbiased estimator of $\partial F(\theta)$, formally:*

$$\mathsf{E}_{\xi \sim P_g}[\partial \ell_g(\theta, \xi^g)] \in \partial F(\theta)$$

.

PROOF Using Theorem 14, for sub-gradients of $F(\theta)$ at a given $\theta \in \Theta$ with respect to $g \in \mathcal{G}(\theta)$ we have:

$$\partial F(\theta) = \text{conv}\{\partial R_g(\theta) : g \in \mathcal{G}(\theta)\}.$$

Using Rockafellar (1974) for sub-differential of integral function, and expanding the definition of $R_g(\theta)$, we get:

$$\partial F(\theta) = \text{conv}\{\mathsf{E}_{\xi \sim P_g}[\partial \ell_g(\theta, \xi^g)] : g \in \mathcal{G}(\theta)\}$$

and the result follows. ∎

Note that the lemma holds even if $F(\theta) = \max_{g \in \mathcal{G}}\{R_g(f) - R_g^*\}$ since R_g^* is given for every $g \in \mathcal{G}$. Now using Assumption 1, one can use the primal updates as in (5.30) to solve the main objective as in (5.13).

5.4 Discussion and Future Work

Now that we have introduced our notion of fairness and provided a solution for a simple case of it, in this section, we explore further challenges and possible approaches to them. Usually in statistical learning, we are interested in providing upper bound on sample complexity of a proposed method, meaning, how many examples do we need to minimize the objective and obtain a fair classifier. However, there are several potential savings in terms of sample complexity that are worth exploring.

First, throughout, we assumed that minimum achievable risks R_g^* for each group are given. One way to obtain R_g^* is to use empirical risk minimization (ERM) to approximate the risk minimizer function in $f_g^* \in \mathcal{F}$. That is

$$f_g^* := \arg\min_{f \in \mathcal{F}} R_g(f),$$

can be estimated. For example, for classes of VC dimension V, we can find ϵ-approximately best function in $\mathcal{F}$ with $\mathcal{O}(V/\epsilon^2)$ labeled samples in agnostic case and with $\mathcal{O}(V/\epsilon)$ in realizable case. The question is, is there a cheaper way to estimate R_g^* (the *value* of minimum achievable risk) instead of the risk minimizer itself? A line of work referred to as (active) property testing might give insights to a possible solution (See Blum and Hu (2017); Balcan et al. (2012), also Foster et al. (2019)[Theorem 2]).

The second important challenge is to solve the problem without Assumption 1 in place. That is, how can we estimate (with some bias) the soft-max using less samples. One approach that might work is to use tools in the differential privacy literature. The stability and adaptive composition properties of differential privacy methods would be useful in this context.

It might be possible to use an exponential mechanism to output (an approximation of) $\tilde{g}_j$ using as few samples as possible. To this end, one needs to define a utility function that maps data and output pair of our algorithm to real values. This is essential for designing an exponential mechanism as it will be required for a mechanism to maximize utility function. The idea is to keep a holdout set Ξ of samples that include samples of all groups and is used to find $\tilde{g}_j$ with respect to θ_j. One might be able to use utility guarantees to show that the coordinate picked by this mechanism is sufficiently close to $\tilde{g}_j$.

5.5 Additional Proofs

In the following, we show how to derive the mirror descent updates as described in Proposition 1.

Proof (Proposition 1) We select regularization functions $\omega_f : \Theta \to \mathbb{R}$ as $\omega_\theta(x) = \frac{1}{2}\|x\|_2^2$ and $\omega_\lambda : \Delta \to \mathbb{R}$ as $\omega_\lambda(x) = \sum_{g \in \mathcal{G}} x_g \ln x_g$. Define $z = (\theta, \lambda)$ and for $Z = \Theta \times \Delta$:

$$\omega(z) = \frac{\omega_\theta(\theta)}{2D^2_{\omega_\theta,\Theta}} + \frac{\omega_\lambda(\lambda)}{2D^2_{\omega_\lambda,\Delta}} \tag{5.14}$$

where $D_{\omega,X}$ is defined as:

$$D_{\omega,X} = \left[\max_{z\in X}\omega(z) - \min_{z\in X}\omega(z)\right]^{1/2}$$

To make (5.14) more explicit, let $B = \max_{\theta\in\Theta}\|\theta\|_2^2$. Hence, we can evaluate $D^2_{\omega_\theta,\Theta}$ as:

$$D^2_{\omega_\theta,\Theta} = \max_{\theta\in\Theta}\left[\frac{1}{2}\|\theta\|_2^2\right] - \min_{\theta\in\Theta}\left[\frac{1}{2}\|\theta\|_2^2\right] = \frac{B}{2}$$

It is easy to verify that the maximum and minimum possible values for Shannon entropy of a random variable with k possible values are $\log k$ and 0. Therefore, for $D^2_{\omega_\lambda,\Delta}$ we have:

$$D^2_{\omega_\lambda,\Delta} = \max_{\lambda\in\Delta}\sum_{g\in\mathcal{G}}\lambda_g\ln\lambda_g - \min_{z\in\Delta}\sum_{g\in\mathcal{G}}\lambda_g\ln\lambda_g = \log|\mathcal{G}|$$

So we can rewrite (5.14) as below:

$$\omega(z) = \frac{\omega_\theta(\theta)}{B} + \frac{\omega_\lambda(\lambda)}{2\log|\mathcal{G}|} = \frac{\|\theta\|_2^2}{2B} + \frac{\sum_{g\in\mathcal{G}}\lambda_g\ln\lambda_g}{2\log|\mathcal{G}|} \tag{5.15}$$

Now in order to get to the algorithm, let us define Bregman distance function $V : Z\times Z\to\mathbb{R}_+$ with respect to ω as:

$$V(z,u) = \omega(u) - [\omega(z) + \langle\nabla\omega(z),(u-z)\rangle] \tag{5.16}$$

and prox-mapping $P_z : \mathbb{R}^{n+m}\to Z$ where $m := |\mathcal{G}|$ as:

$$P_z(u) = \arg\min_{z'\in Z}\{\langle u,(z'-z)\rangle + V(z,z')\} \tag{5.17}$$

Lastly, define $\ell_g(\theta,\xi)$ to be the loss of θ over a sample vector $\xi = (\xi^1,\ldots,\xi^m)$ with respect to group g, i.e., $\ell_g(\theta,\xi)$, and $\mathsf{G}_g(\theta,\xi)$ be its stochastic subgradient, then we define:

$$\mathsf{G}(\theta,\lambda,\xi) = \begin{bmatrix}\mathsf{G}_\lambda(\lambda,\xi)\\ \mathsf{G}_\theta(\theta,\xi)\end{bmatrix} = \begin{bmatrix}\sum_{g\in\mathcal{G}}\lambda_g\mathsf{G}_g(\theta,\xi)\\ -\left(\ell_{g_1}(\theta,\xi^1),\ldots,\ell_{g_m}(\theta,\xi^m)\right)\end{bmatrix}. \tag{5.18}$$

Finally, we are ready to introduce the mirror descent update rule for round j as:

$$z_{j+1} = P_{z_j}(\gamma\mathsf{G}(z_j,\xi_j)) \tag{5.19}$$

where we select the first point $z_1 \in Z$ to be the minimizer for $\omega(z)$ on Z.

In order to make the update rule of (5.19) more explicit for our specific problem, let us expand the definition of prox-mapping P_{z_j}:

$$P_{z_j}(\gamma \mathsf{G}(z_j, \xi_j)) = \arg\min_{z' \in Z} \{\langle \gamma \mathsf{G}(z_j, \xi_j), (z' - z_j)\rangle + V(z_j, z')\} \tag{5.20}$$

Recall that we have defined our regularization function $\omega(z)$ as (5.15), which is a combination of squared Euclidean norm and negative Shannon entropy. Denote by V_θ and V_λ prox-functions for the choice of $\omega_\theta(\theta)$ and $\omega_\lambda(\lambda)$, respectively. Formally, for prox-function V_λ (choice of negative Shannon entropy for $\omega(z)$) we have:

$$\begin{aligned} V_\lambda(x, z) &= \sum_{i=1}^n z_i \ln z_i - \sum_{i=1}^n x_i \ln x_i - \sum_{i=1}^n (1 + \ln x_i)(z_i - x_i) \\ &= \sum_{i=1}^n z_i \ln\left(\frac{z_i}{x_i}\right) + x_i - z_i \end{aligned} \tag{5.21}$$

Note that (5.21) in simplex setup can be simplified to $V_\lambda(x, z) = \sum_{i=1}^n z_i \ln(\frac{z_i}{x_i})$, which also is Kullback-Leibler divergence. For the choice of squared Euclidean norm for $\omega(z)$, V_θ becomes:

$$V_\theta(x, z) = \frac{1}{2}\|z\|_2^2 - \frac{1}{2}\|x\|_2^2 - \langle x, (z - x)\rangle = \frac{1}{2}\|x - z\|_2^2 \tag{5.22}$$

It is easy to verify that defining $\omega(z)$ as (5.15), prox-function can be written as:

$$V(z, z') = \frac{V_\theta(z, z')}{2B} + \frac{V_\lambda(z, z')}{2\log|\mathcal{G}|} \tag{5.23}$$

Using (5.21) and (5.22), for (5.20) we get:

$$P_{z_j}(\gamma \mathsf{G}(z_j, \xi_j)) = \arg\min_{z' \in Z} \left\{ \langle \gamma \mathsf{G}(z_j, \xi_j), (z' - z_j)\rangle + \frac{V_\theta(z_j, z')}{2B} + \frac{V_\lambda(z_j, z')}{2\log|\mathcal{G}|}) \right\} \tag{5.24}$$

Using $z_j = (\theta_j, \lambda_j)$ and unpacking G's definition:

$$\begin{aligned} P_{z_j}(\gamma \mathsf{G}(z_j, \xi_j)) = \arg\min_{(\theta', \lambda') \in Z} \Bigg\{ \left\langle \gamma \begin{bmatrix} \sum_{g \in \mathcal{G}} \lambda_{j,g} \mathsf{G}_g(\theta_j, \xi_j) \\ -(\ell_{g_1}(\theta_j, \xi_j), \ldots, \ell_{g_m}(\theta_j, \xi_j)) \end{bmatrix}, \begin{bmatrix} \lambda' - \lambda_j \\ \theta' - \theta_j \end{bmatrix} \right\rangle \\ + \frac{V_\theta(\theta_j, \theta')}{2B} + \frac{V_\lambda(\lambda_j, \lambda')}{2\log|\mathcal{G}|} \Bigg\} \end{aligned} \tag{5.25}$$

which gives us:

$$P_{z_j}(\gamma \mathsf{G}(z_j, \xi_j)) = \underset{(\theta', \lambda') \in Z}{\arg\min} \left\{ \langle \gamma \mathsf{G}(z_j, \xi_j), (\lambda', \theta') \rangle - \langle \gamma \mathsf{G}(z_j, \xi_j), (\lambda_j, \theta_j) \rangle + \frac{V_\theta(\theta_j, \theta')}{2B} + \frac{V_\lambda(\lambda_j, \lambda')}{2 \log |\mathcal{G}|} \right\}. \tag{5.26}$$

Expanding definitions further and removing independent terms, the minimization objective can be simplified as below:

$$P_{z_j}(\gamma \mathsf{G}(z_j, \xi_j)) = \underset{(\theta', \lambda') \in Z}{\arg\min} \left\{ \langle \gamma \mathsf{G}_\lambda(\lambda_j, \xi_j), \lambda' \rangle + \langle \gamma \mathsf{G}_\theta(\theta_j, \xi_j), \theta' \rangle + V_\theta(\theta_j, \theta') + V_\lambda(\lambda_j, \lambda') \right\}. \tag{5.27}$$

which finally becomes:

$$P_{z_j}(\gamma \mathsf{G}(z_j, \xi_j)) = \left(\underset{\theta' \in \Theta}{\arg\min} \left\{ \langle \gamma \mathsf{G}_\theta(\theta_j, \xi_j), \theta' \rangle + \frac{1}{2} \|\theta' - \theta_j\|_2^2 \right\}, \underset{\lambda' \in \Delta}{\arg\min} \left\{ \langle \gamma \mathsf{G}_\lambda(\lambda_j, \xi_j), \lambda' \rangle + \sum_{i=1}^{n} \lambda'_i \ln \left(\frac{\lambda'_i}{\lambda_{j,i}} \right) \right\} \right). \tag{5.28}$$

The first minimization term can be computed with updates as below:

$$[\theta']_i = \theta_{j,i} - \gamma \mathsf{G}_{\theta,i}(\theta_j, \xi_j) \tag{5.29}$$

which is identical to the gradient descent algorithm. It can be re-written as:

$$\theta_{j+1,g} := \theta'_g = \theta_g - \gamma \ell_g(\theta_j, \xi_j^g) \tag{5.30}$$

The second minimization term results in exponential weights algorithm. So the updates would be as below:

$$[\lambda']_i = \frac{\lambda_{j,i} \exp\{-\gamma \mathsf{G}_{\lambda,i}(\lambda_j, \xi_j)\}}{\sum_{k=1}^{n} \lambda_{j,k} \exp\{-\gamma \mathsf{G}_{\lambda,k}(\lambda_j, \xi_j)\}} \tag{5.31}$$

which can be re-written as:

$$\lambda_{j+1,g} := \lambda'_g = \frac{\lambda_{j,g} \exp\{-\gamma \mathsf{G}_g(\theta_j, \xi_j^g)\}}{\sum_{g \in \mathcal{G}} \lambda_g \exp\{-\gamma \mathsf{G}_g(\theta, \xi^g)\}} = \frac{\lambda_{j,g} \exp\{-\gamma \ell_g(\theta_j, \xi_j^g)\}}{\sum_{g \in \mathcal{G}} \lambda_g \exp\{-\gamma \ell_g(\theta_j, \xi_j^g)\}} \tag{5.32}$$

Error Rate Analysis Here we elaborate on our setup for mirror descent algorithm and discuss detail needed for the analysis of error rate to work. Note that we don't show a full analysis here, instead we refer the reader to Nemirovski et al. (2009)[Section 3.2] for this purpose. Define $R_\theta^2 = \frac{D^2_{\omega_\theta,\Theta}}{\sigma_\theta}$ and $R_\lambda^2 = \frac{D^2_{\omega_\lambda,\Delta}}{\sigma_\lambda}$ where σ_x is strong convexity parameter of ω_x. It is easy to verify that for ω_θ selected as squared Euclidean norm, $\alpha_\theta = 1$. Defining ω_λ as negative Shannon entropy, i.e., $\omega_\lambda(\lambda) = \sum_{g\in\mathcal{G}} \lambda_g \ln \lambda_g$ in the simplex setup where $\Delta = \{\lambda \in \mathbb{R}^n : \lambda_g \geq 0, \sum_{i=1}^n \lambda_g = 1\}$, we show that $\alpha_\lambda = 1$. In order to do so, one needs to verify $h^T \nabla^2 \omega_\lambda(\lambda) h \geq \|h\|_1^2$ for each $\lambda \in \Delta$. Expanding the right-hand side, dividing and multiplying each term by $\lambda_i^{1/2}$, and using Cauchy–Schwarz inequality we have:

$$\left[\sum_i (\lambda_i^{-1/2}|h_i|)\lambda_i^{1/2}\right]^2 \leq \left[\sum_i \lambda_i^{-1} h_i^2\right]\left[\sum_i \lambda_i\right]^2 = \left[\sum_i \lambda_i^{-1} h_i^2\right]^2 = h^T \nabla^2 \omega_\lambda(\lambda) h \tag{5.33}$$

Using these values, one can follow the convergence analysis in Nemirovski et al. (2009)[Section 3.2] to get an error rate of $\mathcal{O}(\sqrt{\frac{\ln|\mathcal{G}|}{N}})$, where N is the number of rounds. Note that the numerator is *almost* independent of the number of groups as desired.■

Chapter 6

Conclusion

In this thesis, we studied problems motivated by fairness in machine learning. Our first problem was concerned with a special case of prediction with expert advice where the set of experts can shrink (dying experts). We showed a lower bound (Theorem 5) on the regret of any algorithm for the case where Learner has no information about the order of experts that are going to die, which matched the existing lower bound from sleeping experts (Kleinberg et al. (2010)). However, unlike the more general case of sleeping experts, we devised an algorithm that can obtain optimal regret in a computationally efficient fashion (Theorem 9). This improvement was possible due to the notion of *effective experts* we introduced and analyzed (Theorem 3). We showed that if Learner is informed about the order of experts that are going to die, the number of effective experts can be reduced by a $\sqrt{\log K}$ multiplicative factor. We obtained a lower bound (Theorem 7) and an optimal computationally efficient algorithm for this case as well (Theorem 10).

In Chapter 4, we attempted to modify the approach in Blum et al. (2018) to obtain an any-time fairness guarantee in online learning. Our setup required a lower bound for the cumulative loss of Hedge using a time-varying learning rate. That is, for an exponential weights algorithm that is using a decreasing learning rate $\eta_t = \sqrt{\frac{\ln K}{t}}$ for $t = 1, 2, \ldots, T$, is there a lower bound for its cumulative loss in the form of Question 1? We used elegant tools in mix loss approximation from De Rooij et al. (2014) to obtain such a lower bound. However, our lower bound was not tight enough to be used for a fairness guarantee, leaving Question 1 open.

Finally, in Chapter 5, we devised a new notion of fairness based on the concept of envy for i.i.d. statistical learning. We explain how this new notion might be better than existing ones. We frame the problem as a stochastic saddle point problem and

use mirror descent algorithm to obtain a solution for a simplified case of it. We discuss and give insights to further challenges that this framework can produce.

Bibliography

Maria-Florina Balcan, Eric Blais, Avrim Blum, and Liu Yang. Active property testing. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 21–30. IEEE, 2012.

Solon Barocas and Andrew D Selbst. Big data's disparate impact. *Calif. L. Rev.*, 104: 671, 2016.

Avrim Blum. Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain. *Machine Learning*, 26(1):5–23, 1997.

Avrim Blum and Lunjia Hu. Active tolerant testing. *arXiv preprint arXiv:1711.00388*, 2017.

Avrim Blum and Yishay Mansour. From external to internal regret. *Journal of Machine Learning Research*, 8(Jun):1307–1324, 2007.

Avrim Blum, Suriya Gunasekar, Thodoris Lykouris, and Nati Srebro. On preserving non-discrimination when combining expert advice. In *Advances in Neural Information Processing Systems*, pages 8386–8397, 2018.

Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

Deepayan Chakrabarti, Ravi Kumar, Filip Radlinski, and Eli Upfal. Mortal multi-armed bandits. In *Advances in neural information processing systems*, pages 273–280, 2009.

Steven De Rooij, Tim Van Erven, Peter D Grünwald, and Wouter M Koolen. Follow the leader if you can, hedge if you must. *The Journal of Machine Learning Research*, 15(1):1281–1316, 2014.

Steven de Rooij, Tim van Erven, Peter D Grünwald, and Wouter M Koolen. Follow the leader if you can, hedge if you must. *The Journal of Machine Learning Research*, 15(1):1281–1316, 2014.

Dylan J Foster, Akshay Krishnamurthy, and Haipeng Luo. Model selection for contextual bandits. In *Advances in Neural Information Processing Systems*, pages 14714–14725, 2019.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

Yoav Freund, Robert E Schapire, Yoram Singer, and Manfred K Warmuth. Using and combining predictors that specialize. In *In Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*. Citeseer, 1997.

Eyal Gofer, Nicolo Cesa-Bianchi, Claudio Gentile, and Yishay Mansour. Regret minimization for branching experts. In *Conference on Learning Theory*, pages 618–638, 2013.

Peter D Grünwald and Nishant A Mehta. Fast rates for general unbounded loss functions: From erm to generalized bayes. *Journal of Machine Learning Research*, 21(56):1–80, 2020.

Moritz Hardt, Eric Price, Nati Srebro, et al. Equality of opportunity in supervised learning. In *Advances in neural information processing systems*, pages 3315–3323, 2016.

JB Hiriart-Urruty. Urruty and c. lemaréchal, fundamentals of convex analysis, 2001.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. In *The Collected Works of Wassily Hoeffding*, pages 409–426. Springer, 1994.

Satyen Kale, Chansoo Lee, and Dávid Pál. Hardness of online sleeping combinatorial optimization problems. In *Advances in Neural Information Processing Systems*, pages 2181–2189, 2016.

Faisal Kamiran and Toon Calders. Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, 33(1):1–33, 2012.

Varun Kanade and Thomas Steinke. Learning hurdles for sleeping experts. *ACM Transactions on Computation Theory (TOCT)*, 6(3):11, 2014.

Varun Kanade, H Brendan McMahan, and Brent Bryan. Sleeping experts and bandits with stochastic action availability and adversarial rewards. In *Artificial Intelligence and Statistics*, pages 272–279, 2009.

Robert Kleinberg, Alexandru Niculescu-Mizil, and Yogeshwer Sharma. Regret bounds for sleeping experts and bandits. *Machine learning*, 80(2-3):245–272, 2010.

Nick Littlestone and Manfred K Warmuth. The weighted majority algorithm. *Information and computation*, 108(2):212–261, 1994.

Jaouad Mourtada. On the optimality of the hedge algorithm in the stochastic regime. *J. Mach. Learn. Res.*, 20:83–1, 2019.

Jaouad Mourtada and Odalric-Ambrym Maillard. Efficient tracking of a growing number of experts. In *International Conference on Algorithmic Learning Theory*, pages 517–539, 2017.

Cecilia Munoz, Megan Smith, and DJ Patil. *Big data: A report on algorithmic systems, opportunity, and civil rights*. Executive Office of the President, 2016.

Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

Gergely Neu and Michal Valko. Online combinatorial optimization with stochastic decision sets and adversarial losses. In *Advances in Neural Information Processing Systems*, pages 2780–2788, 2014.

Francesco Orabona and Dávid Pál. Optimal non-asymptotic lower bound on the minimax regret of learning with expert advice. *arXiv preprint arXiv:1511.02176*, 2015.

Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. On fairness and calibration. In *Advances in Neural Information Processing Systems*, pages 5680–5689, 2017.

R Tyrrell Rockafellar. *Conjugate duality and optimization*. SIAM, 1974.

Samira Samadi, Uthaipon Tantipongpipat, Jamie H Morgenstern, Mohit Singh, and Santosh Vempala. The price of fair pca: One extra dimension. In *Advances in Neural Information Processing Systems*, pages 10976–10987, 2018.

Hamid Shayestehmanesh, Sajjad Azami, and Nishant A Mehta. Dying experts: Efficient algorithms with optimal regret bounds. In *Advances in Neural Information Processing Systems*, pages 9983–9992, 2019.

Michel Talagrand. Regularity of infinitely divisible processes. *The Annals of Probability*, 21(1):362–432, 1993.

Michel Talagrand. *The generic chaining*, volume 154. Springer, 2005.

Tim van Erven, Wouter M Koolen, Steven de Rooij, and Peter Grünwald. Adaptive Hedge. In *Advances in Neural Information Processing Systems*, pages 1656–1664, 2011.

Vladimir Vovk. Aggregating strategies. In *Proceedings of the third annual workshop on Computational learning theory*, pages 371–386. Morgan Kaufmann Publishers Inc., 1990.

Vladimir Vovk. A game of prediction with expert advice. *Journal of Computer and System Sciences*, 56(2):153–173, 1998.

Robert Williamson and Aditya Menon. Fairness risk measures. In *International Conference on Machine Learning*, pages 6786–6797, 2019.