Theory and Algorithms for Dynamic and Adaptive Online Learning

by

Scott Yang

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Mathematics

New York University

May, 2017

Professor Mehryar Mohri

Dedication

To my best friend and wife Annie.

Acknowledgements

First and foremost, I would like to acknowledge my grandparents and parents for instilling the value of education and hard work in me at an early age. My grandparents played a key role in my upbringing, and, despite not having had the opportunity to finish even elementary school themselves, have always encouraged me and made sure that I took my studies seriously. I also must thank my parents, Roger and Agnes Yang, for their unwavering support in all of my endeavors and for always trying to play any part they could can to help me accomplish my dreams. I would not have even entertained the idea of pursuing a doctorate without the key values that have been instilled by my most important role models.

The decision to enroll in graduate school would also have never been possible without Michael Damron, who, as a graduate student, advised me on an undergraduate research project. While he started out as an advisor, he quickly became an extraordinary mentor and friend who had a profound impact on me and has come to shape much of my perspective and approach to mathematics.

I also cannot thank my research advisor, Mehryar Mohri, enough, for the invaluable role he has played in my growth and development during my time at graduate school. He has not only been a boundless source of knowledge on all things related to machine learning, but has also acted as a role model in all aspects of being a researcher. It is through him that I learned the importance of playing an active role in the research community, delivering clear and engaging presentations, and writing papers with both clarity and care. He has also taught me that completing a Ph.D. is as much about the journey as it is about the destination. I will be forever indebted to him for all of his time, effort, and support these past years.

My research group at Courant, which has consisted of Giulia DeSalvo, Andrés

Muñoz Medina, Vitaly Kuznetsov, Dmitry Storcheus, and Ningshan Zhang over the years, has also been an incredible source of motivation and support. They have together played a significant role in my development as a machine learning researcher. Many of these individuals have been collaborators, and all of them have been and will continue to be friends.

During my graduate school years, I have also had the opportunity to engage in two exciting and fulfilling internships at Google Research. I would like to express my sincere gratitude to my internship hosts, Umar Syed and Rajhans Samdani, for taking a chance on me as their intern, providing me with exciting projects, and affording me their time and guidance over those summers. I gained an incredible wealth of information and experience from them for which I am extremely grateful.

I would also like to thank my thesis committee, Robert Kohn, Esteban Tabak, Corinna Cortes, and Satyen Kale, for their time and effort in assisting in the completion of my thesis. They are all extremely accomplished researchers and serve as a great source of inspiration and admiration.

Finally, I would like to thank my wife, Annie Wei, for her unconditional love and support over the years. We started dating before I ever even entertained the idea of graduate school, and I cannot show her enough appreciation and gratitude for her patience and commitment to me. She has been a constant bright source of encouragement and affection, without which I would have never been able to complete my graduate studies.

In summary, this thesis, as well as the rest of my work in graduate school, would not have been possible without all of the wonderful individuals mentioned above, as well as so many more. I am extremely grateful for all the friends and colleagues that I have made during the course of this incredible journey.

Abstract

Online learning is a powerful and flexible model for sequential prediction. In this model, algorithms process one sample at a time with an update per iteration that is often computationally cheap and simple to implement. As a result, online learning algorithms have become an attractive solution for modern machine learning applications with very large data sets.

The standard benchmark adopted in online learning is external regret, which is the difference between the cumulative loss of an algorithm and that of the best static model in hindsight. Remarkably, there exist online learning algorithms for which the average external regret converges to zero, even when the learner receives only limited information about the environment or the data-generating distribution. However, this measure of performance may not be useful when the data is non-stationary and no static model admits a small loss. It may also be inadequate when a learner's algorithm is not robust and a simple perturbation of the algorithm's predictions could have been substantially more effective. By construction, online learning algorithms that are designed to minimize external regret cannot be expected to perform well in dynamic environments.

Most online learning algorithms are also designed with worst-case guarantees in mind, so that they will guarantee a minimum level of performance. This often makes their updates and predictions overly conservative, so that they are unable to adapt to and learn faster from easier data. Finally, while model selection has been well-developed in the offline setting using ideas such as structural risk minimization, its treatment is lacking in the online learning setting.

This dissertation addresses all the issues just mentioned and presents online learning algorithms that achieve more compelling guarantees by adjusting for

dynamic environments and *adapting* to the given data. In particular, it develops:

- a novel and flexible framework that can be used to design efficient online algorithms with strong guarantees against non-stationary and evolving benchmarks;
- a new algorithm that can guarantee the robustness of the learners' predictions and that is motivated by insightful game theoretic considerations;
- a new and general framework for online convex optimization that can leverage a learner's predictions and tune an algorithm's performance to adaptively optimize for those predictions;
- a new theoretical analysis and algorithmic design for model selection applied to online learning that is consistent with the offline scenario.

Contents

List of Figures

List of Appendices

Introduction

Online learning is a powerful and flexible model for sequential prediction. Online learning algorithms represent key tools in modern machine learning. They are flexible, computationally efficient, and can be used to solve a variety of problems in machine learning, including those arising in classification, regression, ranking, and probabilistic inference.

In the online learning model, algorithms process one sample at a time with an update per iteration that is often computationally cheap and easy to implement. As a result, online learning algorithms can be substantially more efficient both in time and space than standard batch learning algorithms, and they have become an attractive solution for large-scale machine learning problems.

Online learning solutions have been designed for many applications where one must make real-time decisions with sequential data. These include problems such as click-through-rate prediction for advertising placement [McMahan et al., 2013, He et al., 2014], online recommendation systems [Das et al., 2007], weather prediction [Monteleoni et al., 2013], and large-scale optimization [Recht et al., 2011].

The standard model for online learning is a sequential prediction problem proceeding over a number of rounds $t = 1, 2, \ldots, T$. At each round t, the learner's algorithm $\mathcal{A}$ chooses an action $x_t \in \mathcal{X}$ from some space, receives a loss function l_t from an adversary, and incurs the loss $l_t(x_t)$. The algorithm's performance is measured by its cumulative loss: $\sum_{t=1}^{T} l_t(x_t)$, and the standard benchmark that is typically adopted is the *external regret* [Cesa-Bianchi and Lugosi, 2006], which is the difference between the cumulative loss of the learner's algorithm and the

cumulative loss of the best static action chosen in hindsight:

$$\mathrm{Reg}_T(\mathcal{A}) = \sum_{t=1}^{T} l_t(x_t) - \min_{x \in \mathcal{X}} \sum_{t=1}^{T} l_t(x).$$

Important special cases of this model include the setting of *prediction with expert advice* [Cesa-Bianchi and Lugosi, 2006]. Here, the action space $\mathcal{X} = \{1, 2 \ldots, N\}$ is a discrete set corresponding to a finite set of experts. At each round, the learner decides to follow the advice of a particular expert, and the algorithm's cumulative loss is compared against the cumulative loss of the best expert (which is not known a priori). A classical algorithm for solving the prediction with expert advice problem is the RandomizedWeightedMajority (RWM) algorithm [Littlestone and Warmuth, 1994]. In this algorithm, the learner keeps track of the cumulative loss of each expert: $L_{t,i} = \sum_{s=1}^{t} l_t(i)$, and at each round, the learner samples an expert at random according to the distribution: $\mathsf{p}_t[i] \propto \exp -\eta L_{t,i}$. If the losses at each round l_t are bounded by some fixed constant L, and η is chosen to be $\frac{1}{\sqrt{LT}}$, then the *expected regret* of this algorithm will be:

$$\mathrm{Reg}_T(\mathrm{RWM}) = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{x_t \sim \mathsf{p}_t} [l_t(x_t)] - \min_{x \in \mathcal{X}} \sum_{t=1}^{T} l_t(i) \leq \mathcal{O}\left(\sqrt{T}\right).$$

It is remarkable that there exist algorithms for which the average external regret converges to zero, especially since the learner assumes only limited information about the environment or the data-generating distribution. However, this measure of performance may not be useful when the data is non-stationary and no static model admits a small loss. For example, it may be the case that different experts become more effective at different times and that no single expert performs well over the entire data.

External regret may also be inadequate when a learner's algorithm is not robust and a simple perturbation of the algorithm's predictions could have been substantially more effective. For instance, an algorithm that could have performed much better by displaying ad A every time it displayed ad B would have poor understanding of its available actions or experts. Unfortunately, by construction, online learning algorithms that are designed to minimize external regret cannot be expected to perform well in dynamic environments.

Another important variant of online learning is the *online convex optimization (OCO)* model [Shalev-Shwartz, 2012]. In this setting, the action space $\mathcal{X}$ is assumed to be a compact convex set, and the loss functions l_t are also assumed to be convex. If the loss functions are static over time, then by Jensen's inequality, the average cumulative loss of the learner's actions is bounded below by the loss of the average action (which is itself an action due to convexity of the action space):

$$\sum_{t=1}^{T} l\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) \leq \frac{1}{T}\sum_{t=1}^{T} l(x_t).$$

Thus, online convex optimization algorithms can be easily converted into convex optimization methods, and regret bounds for online convex optimization immediately yield convergence guarantees for convex optimization. Similarly, standard methods in convex optimization, such as the gradient descent algorithm, have direct analogues in online convex optimization, in the form of OnlineGradientDescent. In the online version, the learner applies the update rule:

$$x_{t+1} \leftarrow \operatorname*{argmin}_{x \in \mathcal{X}} \eta g_t^\top x + \|x - x_t\|_2^2,$$

where $g_t \in \partial l_t(x_t)$ is an element of the subgradient of the loss function l_t at x_t. If l_t

L-Lipschitz and $\mathcal{X}$ has diameter at most D^2, then OnlineGradientDescent with learning rate $\eta = \frac{D}{\sqrt{LT}}$ will guarantee regret at most:

$$\mathrm{Reg}_T(\textsc{OnlineGradientDescent}) = \mathcal{O}(DL\sqrt{T}).$$

Similarly, there are also online convex optimization generalizations of the MirrorDescent or DualAveraging algorithms, in the form of OnlineMirrorDescent and Follow-the-Regularized-Leader.

Most online learning algorithms, such as the ones above, are designed with worst-case guarantees in mind. They guarantee a minimum level of performance. However, this often makes their updates and predictions overly conservative, so that they are unable to adapt to and learn faster from easier data.

Apart from the algorithmic approaches discussed so far, there have also been purely theoretic analyses of online learning problems. By interpreting the online learning model as a repeated game between a learner and an adversary where the learner selects distributions $\mathsf{p}_t \in \Delta(\mathcal{X})$ over actions and the adversary selects loss functions from a function space $\mathcal{L}$, it becomes natural to consider the *minimax* value of this game

$$\mathcal{V}_T(\mathcal{X}, \mathcal{L}) = \inf_{\mathsf{p}_1 \in \Delta(\mathcal{X})} \sup_{l_1 \in \mathcal{L}} \mathop{\mathbb{E}}_{x_1 \sim \mathsf{p}_1} \ldots \inf_{\mathsf{p}_T \in \Delta(\mathcal{X})} \sup_{l_T \in \mathcal{L}} \mathop{\mathbb{E}}_{x_T \sim \mathsf{p}_T} \left[\sum_{t=1}^{T} l_t(x_t) - \inf_{x \in \mathcal{X}} \sum_{t=1}^{T} l_t \right].$$

Note that a game whose minimax value grows sublinearly in T implies the existence of a randomized strategy with sublinear regret. Moreover, it is possible to define a complexity measure over the action space, the *sequential Rademacher complexity* $\mathfrak{R}_T(\mathcal{X})$, such that the value of the game is bounded by twice the sequential Rademacher complexity [Rakhlin et al., 2010]: $\mathcal{V}_T(\mathcal{X}, \mathcal{L}) \leq 2\mathfrak{R}_T(\mathcal{X})$. Thus, sublinear

growth rates of the sequential Rademacher complexity imply the existence of strategies with sublinear regret.

While this result is remarkable in its generality, it does not capture the problem of model selection in the online learning setting, where the learner has access to multiple action sets $\{\mathcal{X}_i\}_{i=1}^p$ and must find an efficient way choose among them. This problem has been addressed in the batch learning setting in the form of *structural risk minimization* [Vapnik, 1992] and *voted risk minimization* [Cortes et al., 2014], but it has not been analyzed in the online learning model.

This thesis addresses all of the issues and settings discussed above, and it presents online learning algorithms with theoretical guarantees that are more compelling in *dynamic* environments and that *adapt* to the given data.

In Chapter 1, it develops a novel and flexible framework that can be used to design efficient online algorithms with strong guarantees against non-stationary and dynamic benchmarks. In Chapter 2, it presents a new family of algorithms that can guarantee the robustness of the learners' predictions and that is motivated by insightful game theoretic considerations. In Chapter 3, it introduces a new and general framework for online convex optimization that can leverage a learner's predictions and tune an algorithm's performance to adaptively optimize for those predictions. In Chapter 4, it presents a new theoretical analysis and algorithmic design for model selection applied to online learning that is consistent with the offline scenario.[1]

[1]Part of this thesis is based on work in [Mohri and Yang, 2014, 2016a,b]

Chapter 1

Online Learning with Expert Automata

In this chapter, we consider a general framework of online learning with expert advice where the regret is defined with respect to a competitor class defined by a weighted automaton over sequences of experts. Our framework covers several problems previously studied, in particular that of competing against k-shifting experts.

We give a series of algorithms for this problem, including an automata-based algorithm extending the weighted-majority algorithm and more efficient algorithms based on the notion of failure transitions. We further present efficient algorithms based on a compact approximation of the competitor automaton, in particular efficient n-gram models obtained by minimizing the Rényi divergence, and we present an extensive study of the approximation properties of such models. We also extend our algorithms and results to the framework of sleeping experts. Finally, we describe the extension of our approximation methods to online convex optimization

and a general mirror descent setting.

Within the online learning framework, the setting of prediction with expert advice has received widespread attention [Littlestone and Warmuth, 1994, Cesa-Bianchi and Lugosi, 2006, Cesa-Bianchi et al., 2007]. In this setting, the algorithm maintains a distribution over a set of experts, or selects an expert from an implicitly maintained distribution. At each round, the loss assigned to each expert is revealed. The algorithm incurs the expected loss over the experts and then updates its distribution on the set of experts. The objective of the learner is to minimize his expected regret, which is defined as the cumulative loss of the algorithm minus the cumulative loss of the best expert chosen in hindsight.

However, this benchmark is only significant when the best expert is expected to perform well. When this is not the case, then the learner may still play poorly. As an example, it may be that no single baseball team has performed well over all seasons in the past few years. Instead, different teams may have dominated over different time periods. This has led to a definition of regret against the best sequence of experts with k shifts in the seminal work of Herbster and Warmuth [1998] on *tracking the best expert*. The authors showed that there exists an efficient online learning algorithm for this setting with favorable regret guarantees.

This work has subsequently been improved to account for broader expert classes [Gyorgy et al., 2012], to deal with unknown parameters [Monteleoni and Jaakkola, 2003], and has been further generalized [Cesa-Bianchi et al., 2012, Vovk, 1999]. Another approach for handling dynamic environments has consisted of designing algorithms that guarantee small regret over any subinterval during the course of play. This notion coined as *adaptive regret* by Hazan and Seshadhri [2009] has been subsequently strengthened and generalized [Daniely et al., 2015, Adamskiy et al.,

2012]. Remarkably, it was shown by Adamskiy et al. [2012] that the algorithm designed by Herbster and Warmuth [1998] is also optimal for adaptive regret. Koolen and de Rooij [2013] described a Bayesian framework for online learning where the learner samples from a distribution of expert sequences and predicts according to the prediction of that expert sequence. They showed how the algorithms designed for k-shifting regret, e.g. [Herbster and Warmuth, 1998, Monteleoni and Jaakkola, 2003], can be interpreted as specific priors in this formulation. There has also been work deriving guarantees in the bandit setting when the losses are stochastic [Besbes et al., 2014, Wei et al., 2016].

The general problem of online convex optimization in the presence of non-stationary environments has also been studied by many researchers. One perspective has been the design of algorithms that maintain a guarantee against sequences that do not vary too much [Mokhtari et al., 2016, Shahrampour and Jadbabaie, 2016, Jadbabaie et al., 2015, Besbes et al., 2015]. Another assumes that the learner has access to a dynamical model that is able to capture the benchmark sequence [Hall and Willett, 2013]. György and Szepesvári [2016] reinterpreted the framework of Hall and Willett [2013] to recover and extend the results of Herbster and Warmuth [1998].

In this paper, we significantly generalize the framework just described and consider prediction with expert advice in a setting where the learner's cumulative loss is compared against that of sequences represented by an *arbitrary weighted family of sequences*. We model this family using a weighted finite automaton (WFA). This strictly generalizes the notion of k-shifting regret and extends it to the notion of regret against a WFA.

Measuring regret against an automaton is both natural and flexible. In fact, it

may often be sensible to *learn* the set of competitor sequences using data before competing against it. For instance, the competitor automaton could be a language model trained over best sequences of baseball teams in the past. Moreover, the competitor automaton could be learned and reset incrementally. After each epoch, we could choose to learn a new competitor model and seek to perform well against that.

We show that not only it is possible to achieve favorable regret against a WFA but that there exist *computationally efficient* algorithms to achieve that. We give a series of algorithms for this problem. Our first algorithm (Section 1.5) is an automata-based algorithm extending the weighted-majority algorithm and uses automata operations such as composition and shortest-distance; its computational cost is exponentially better than that of a naïve method.

We further present efficient algorithms based on a compact approximation of the competitor automaton (Section 1.6), in particular efficient n-gram models obtained by minimizing the Rényi divergence, and present an extensive study of the approximation properties of such models. We also show how existing algorithms for minimizing k-shifting regret can be recovered by learning a Maximum-Likelihood bigram language model over the k-shifting competitor automaton. To the best of our knowledge, this is the first instance of recovering the algorithms of Herbster and Warmuth [1998] by way of solely focusing on minimizing the k-shifting regret. Since approximating the competitor automaton is subject to a trade-off between computational efficiency and approximation accuracy, we also design a model selection algorithm adapted to this problem.

We further improve that algorithm by using the notion of failure transitions (φ-transitions) for a more compact and therefore more efficient automata representation.

Here, we design a new algorithm (Section 1.8) that can convert any weighted finite automaton into a weighted finite automaton with φ-transitions (φ-WFA). We then extend the classical composition and shortest-distance algorithms for WFAs to the setting of φ-WFAs. The shortest-distance algorithm is designed by extending the probability semiring structure of the φ-WFA to that of a ring. We show that if the number of consecutive φ-transitions is not too large, then these algorithms have a computational complexity that is comparable to those for standard WFAs. At the same time, our conversion algorithm can dramatically reduce the size of a WFA.

We then extend the results above to the sleeping expert setting [Freund et al., 1997], where the learner may not have access to advice from every expert at each round (Section 1.10). Finally, we extend the ideas for prediction with expert advice to online convex optimization and a general mirror descent setting (Section 1.11). Here, we describe a related framework that parallels the previous discussion and also recovers existing algorithms for k-shifting regret.

1.1 Preliminaries

A weighted finite automaton (WFA) is a finite automaton whose transitions, initial, and final states additionally carry some weights. For our purpose, the weights belong to the probability semiring $(\mathbb{R}_+ \cup \{+\infty\}, +, \times, 0, 1)$:[1] we use multiplication to compute path weights by taking the product of the transition weights, and use addition to compute the weight of any string x by taking the sum of the weights of all accepted paths labeled with x. We assume that all automata are *deterministic*,

[1] For numerical stability, in practice, the computations should be performed in the log-semiring: $(\mathbb{R} \cup \{-\infty, +\infty\}, \oplus_{\log}, +, +\infty, 0)$, where $\oplus_{\log}$ is defined by $x \oplus_{\log} y = -\log(e^{-x} + e^{-y})$, which is isomorphic to the probability semiring under the mapping $w \mapsto -\log(w)$.

so that for any state q and label a, there exists at most one transition leaving q labeled with a.

For a WFA $\mathcal{A}$, we denote by $Q_{\mathcal{A}}$ its finite set of states, by $I_{\mathcal{A}} \in Q_{\mathcal{A}}$ its initial states, by $F_{\mathcal{A}} \subseteq Q_{\mathcal{A}}$ its final states, and by $E_{\mathcal{A}}$ its finite set of transitions, which are elements of $Q_{\mathcal{A}} \times \Sigma \times \mathbb{R}_+ \times Q_{\mathcal{A}}$, where Σ is a finite alphabet. We also denote by $\lambda\colon I_{\mathcal{A}} \to \mathbb{R}_+$ an initial weight function, and by $\rho\colon F_{\mathcal{A}} \to \mathbb{R}_+$ a final weight function. This allows us to define a WFA as a 7-tuple: $\mathcal{A} = (\Sigma, Q_{\mathcal{A}}, I_{\mathcal{A}}, F_{\mathcal{A}}, E_{\mathcal{A}}, \lambda_{\mathcal{A}}, \rho_{\mathcal{A}})$.

Given a transition $e \in E_{\mathcal{A}}$, we denote by $w[e] \in \mathbb{R}_+$ its weight. A path π in $\mathcal{A}$ is an element of $E_{\mathcal{A}}^*$ with consecutive transitions. We denote by $\mathrm{src}[\pi]$ the source or origin of the path and by $\mathrm{dest}[\pi]$ its destination. The weight of a path in $\mathcal{A}$ is defined to be the product of the weights of its constituent transitions, thus the weight of path $\pi = e_1 \dots e_n$ is $w[\pi] = \prod_{i=1}^{n} w[e_i]$.

For any string $x \in \Sigma^*$, let $P(I_{\mathcal{A}}, x, F_{\mathcal{A}})$ denote the set of paths accepted by $\mathcal{A}$ and labeled with x. A WFA $\mathcal{A}$ over an alphabet Σ defines a function, which we abusively also denote by $\mathcal{A}$, mapping a string $x \in \Sigma^*$ to $\mathbb{R}_+ \cup \{+\infty\}$ and defined as follows:

$$\forall x \in \Sigma^*, \quad \mathcal{A}(x) = \sum_{\pi \in P(I_{\mathcal{A}}, x, F_{\mathcal{A}})} w[\pi] \lambda_{\mathcal{A}}(\mathrm{src}[\pi]) \rho_{\mathcal{A}}(\mathrm{dest}[\pi]), \tag{1.1}$$

where, by convention, $\mathcal{A}(x) = 0$ if $P(I_{\mathcal{A}}, x, F_{\mathcal{A}}) = \varnothing$. Thus, $\mathcal{A}(x)$ is the sum of the weights of all paths accepted by $\mathcal{A}$ and labeled with string x, multiplied by the weight of the initial and final states. In the probability semiring, the order of the terms in the sum does not matter, and the quantity is well defined [Mohri, 2009]. The *size of a WFA* is denoted by $|\mathcal{A}|$ and defined as the sum of the number of states and the number of transitions of $\mathcal{A}$: $|\mathcal{A}| = |Q_{\mathcal{A}}| + |E_{\mathcal{A}}|$.

For any state $q \in Q$, we use $E[q]$ to denote the outgoing transitions of q, and more generally, $E[Q'] = \cup_{q \in Q'} E[q]$ to denote the set of transitions leaving states in $Q' \subseteq Q$. Given a label $a \in \Sigma$, we also denote by $E[q, a]$ the set of outgoing transitions of q with label a, and by $E[Q', a]$ the set of transitions leaving states in Q' with label a. Similarly, we let $E^R[q]$ denote the incoming transitions of q and define, $E^R[Q'] = \cup_{q \in Q'} E^R[q]$, $E^R[q, a]$, and $E^R[Q', a]$ analogously.

For any state $q \in Q$ and any symbol $a \in \Sigma$, we denote by $\delta[q, a]$ the set of states reachable from q by reading label a. By overloading notation, this implies that $\delta[q, a] = \text{dest}[E[q, a]]$. We also define $\delta[Q', a] = \cup_{q \in Q'} \delta[q, a]$, $\delta[q] = \cup_{a \in \Sigma} \delta[q, a]$ and $\delta[Q'] = \cup_{q \in Q'} \delta[q]$. Similar to $E[\cdot]$ and $E^R[\cdot]$, we also denote by $\delta^R[q, a]$ the set of states with transitions labeled by a that lead into state q, along with $\delta^R[Q', a]$, $\delta^R[q]$, and $\delta^R[Q']$ by analogy.

Given any automaton $\mathcal{A}$, we denote by $\mathcal{A}^R$ its reversal, that is the automaton derived from $\mathcal{A}$ by reverting the direction of each transition and by swapping initial and final state sets. Thus, the strings accepted by $\mathcal{A}^R$ are the mirror images of those accepted by $\mathcal{A}$. Moreover, the corresponding paths share the same weight.

1.2 Learning problem

We consider the setting of prediction with expert advice, which can be seen as a sequential game over T rounds. Let $\Sigma = \{a_1, a_2, \ldots, a_N\}$ be a set of N experts. At each round t, the learner specifies a probability distribution p_t over the set of experts Σ, samples an expert $i_t \sim \mathsf{p}_t$ from the distribution, receives the loss vector $\mathbf{l}_t \in [0, 1]^N$, and incurs the specific loss $l_t[i_t]$. The learner's goal is to minimize her regret, that is the difference between the cumulative expected loss of the learner's

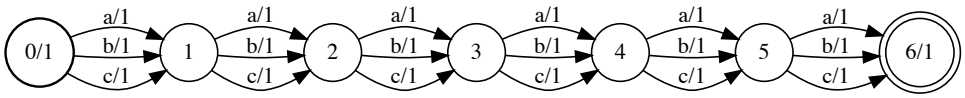


Figure 1.1: Illustration of $\mathcal{S}_T$, for $\Sigma = \{a, b, c\}$.

algorithm and the best loss corresponding to some benchmark competitor class. The standard benchmark in this setting is that of static experts, leading to the following notion of (expected) regret:

$$\mathrm{Reg}_T(\mathcal{A}, \Sigma) = \max_{a \in \Sigma} \sum_{t=1}^{T} \mathsf{p}_t \cdot \mathbf{l}_t - \sum_{t=1}^{T} l_t[a]. \tag{1.2}$$

However, static experts may be too weak a benchmark in many settings where the data is potentially non-stationary and where no single expert is accurate or effective during all time segments of the game. Thus, instead, we consider the regret of the learner against a set of sequences of experts represented by an automaton. Specifically, let $\mathcal{C}$ be a weighted finite automaton (WFA) over the semiring $(\mathbb{R}_+, +, \times, 0, 1)$ with the alphabet Σ. $\mathcal{C}$ accepts a (potentially infinite) set of sequences of experts of arbitrary length. For instance, $\mathcal{C}$ may be the set of expert sequences that use each expert once, or the set of sequences that use at most two experts. Let $\mathcal{S}_T$ be the weighted finite automaton that accepts all sequences of length T and assigns a weight of one to every transition. Specifically, we define $Q_{\mathcal{S}_T} = \{t\}_{t \in [0,T]}$, $I_{\mathcal{S}_T} = \{0\}$, $F_{\mathcal{S}_T} = \{T\}$, and $E_{\mathcal{S}_T} = \{(t-1, k, 1, t)\}_{t \in [1,T], k \in \Sigma}$. This automaton is illustrated in Figure 1.1. Notice that $\mathcal{S}_T$ is acyclic.

Then, $\mathcal{C}_T = \mathcal{C} \cap \mathcal{S}_T$, where $\cap$ denotes the intersection operation between automata,

will define the set of expert sequences that our learning algorithm will compare against. Since $\mathcal{C}_T$ accepts only a finite number of sequences, we can normalize the automaton $\mathcal{C}_T$ so that the total weight over all expert sequences of length T is one. Let $\mathsf{p}_{\mathcal{C}_T}$ be the probability distribution over sequences accepted by $\mathcal{C}_T$ defined by: $\mathsf{p}_{\mathcal{C}_T}(z_1^T) = \frac{w(z_1^T)}{\sum_{\tilde{z}_1^T \in \mathcal{C}_T} w(\tilde{z}_1^T)}$. We now define the *weighted regret* of algorithm $\mathcal{A}$ against automaton $\mathcal{C}$ up to time T as:

$$\mathrm{Reg}_T(\mathcal{A}, \mathcal{C}) = \max_{z_1^T \in \mathcal{C}_T} \sum_{t=1}^{T} \mathsf{p}_t \cdot \mathbf{l}_t - \left(\sum_{t=1}^{T} l_t[z_t] - \log \left(\mathsf{p}_{\mathcal{C}_T}(z_1^T) |\mathcal{C}_T| \right) \right), \tag{1.3}$$

where $|\mathcal{C}_T|$ is the number of paths accepted by $\mathcal{C}_T$. This notion of regret uses sequences in $\mathcal{C}_T = \mathcal{C} \cap \mathcal{S}_T$ as a benchmark, and penalizes the performance of an algorithm against different expert sequences based on the learner's prior characterization of the importance of each sequence. $|\mathcal{C}_T|$ is a normalization factor added so that if the automaton's weight is uniform over all sequences, then the weighted regret is consistent with the standard notion of regret. To the best of our knowledge, previous work has not considered prediction with expert advice with a benchmark loss that differs over competitors.

To compare our work with previous results, we also define the *unweighted regret* of algorithm $\mathcal{A}$ against automaton $\mathcal{C}$ up to time T as:

$$\mathrm{Reg}_T^0(\mathcal{A}, \mathcal{C}) = \max_{z_1^T \in \mathcal{C}_T} \sum_{t=1}^{T} \mathsf{p}_t \cdot \mathbf{l}_t - \sum_{t=1}^{T} l_t[z_t]. \tag{1.4}$$

Note that the automaton $\mathcal{C}$ can still have arbitrary weights, although they play no role in this notion of regret. When $\mathcal{C}$ has uniform weights, then the unweighted regret and weighted regret are identical.

If we associate with Σ the automaton accepting constant sequences and with

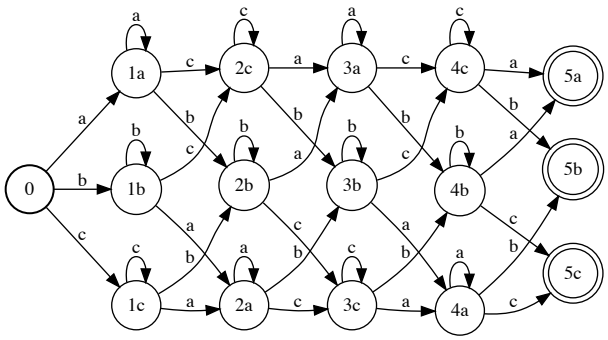


Figure 1.2: An automaton accepting the k-shifting experts competitor class.

uniform weights over all sequences, then this definition of regret is consistent with the expression $\mathrm{Reg}_T(\mathcal{A}, \Sigma)$ in equation 1.2.

As a running example, the k-shifting expert problem can be cast as the automaton $\mathcal{C}_{k\text{-shift}}$ with uniform weights given by Figure 1.2. Then, the (weighted or unweighted) regret of an algorithm $\mathcal{A}$ against $\mathcal{C}_{k\text{-shift}}$ coincides with the definition of k-shifting regret studied by Herbster and Warmuth [1998]:

$$\mathcal{C}_{k\text{-shift}} = \left\{ \mathbf{z} \in \Sigma^* \colon \sum_{t=0}^{\infty} 1_{z_{t+1} \neq z_t} = k \right\}.$$

1.3 Naïve algorithm

A well-known algorithm for minimizing static regret in the prediction with expert advice setting is the weighted majority algorithm [Littlestone and Warmuth, 1994]. The algorithm maintains a distribution over experts that is proportional to the exponential weight of the cumulative loss of the experts: $\mathsf{p}_t[i] \propto e^{-\eta \sum_{s=1}^{t} l_t[i]}$ for

Algorithm 1: PATH-BASED WEIGHTED MAJORITY(PBWM).

Algorithm: PBWM($\mathcal{C} \cap \mathcal{S}_T$, $\tilde{v}_1$),
where $\tilde{v}_1 \in \mathbb{R}_+^K$ is a vector of initial path weights.

$\boldsymbol{\pi} \leftarrow \mathcal{C} \cap \mathcal{S}_T$
$K \leftarrow |\mathcal{C} \cap \mathcal{S}_T|$
$N \leftarrow |\Sigma|$
for $j = 1$ *to* K **do**
  $\tilde{p}_{1,j} \leftarrow \frac{\tilde{v}_{1,j}}{\sum_{i=1}^K \tilde{v}_{1,i}}$
for $j = 1$ *to* N **do**
  $p_{1,j} \leftarrow \sum_{i \in [1,K]:\, \pi_{i,1}=j} \tilde{p}_{1,i}$
for $t = 1$ *to* T **do**
  $i_t \leftarrow$SAMPLE(p_t)
  PLAY(i_t)
  RECEIVE($\mathbf{l}_t$)
  for $j = 1$ *to* K **do**
    $\tilde{v}_{t+1,j} \leftarrow \tilde{v}_{t,j} e^{-\eta l_t[\pi_{j,t}]}$
    $\tilde{p}_{t+1,j} \leftarrow \frac{\tilde{v}_{t+1,j}}{\sum_{i=1}^K \tilde{v}_{t+1,i}}$
  for $j = 1$ *to* N **do**
    $p_{t+1,j} \leftarrow \sum_{i \in [1,K]:\, \pi_{i,t}=j} \tilde{p}_{t+1,i}$

some $\eta > 0$.

We can extend this idea to path experts and design an algorithm that minimizes the unweighted and weighted regret. Specifically, suppose we enumerate the paths accepted by $\mathcal{C}_T$ by $\{\pi_j\}_{j=1}^K$. At each round t, each path chooses an expert $\pi_{j,t} \in \Sigma$, and can be attributed the loss $l_t[\pi_{j,t}]$. Thus, we can define a weight distribution over the experts using the formula: $\tilde{\mathsf{p}}_t[j] \propto e^{-\eta \sum_{s=1}^t l_s[\pi_{j,s}]}$. We can then convert this into a distribution over experts using the formula: $\mathsf{p}_t[i] \propto \sum_{j=1}^K \tilde{p}_t[j] 1_{\pi_{j,t}=i}$. We call this algorithm PATH-BASED WEIGHTED MAJORITY (PBWM). Its pseudocode is presented in Algorithm 1. The following guarantee holds for the regret of PBWM, which is proven in Appendix A.1.

Theorem 1.1 (Regret Bounds for Path-Based Weighted Majority)**.** *Let* $\tilde{v}_{1,j} = 1$

for every $j \in [K]$ be the initial path weights in the PBWM *algorithm. Then, the unweighted regret of* PBWM *is bounded as follows:*

$$\mathrm{Reg}_T^0(\mathrm{PBWM}, \mathcal{C}) \leq \frac{\eta T}{8} + \frac{1}{\eta} \log(|\mathcal{C} \cap \mathcal{S}_T|).$$

Alternatively, assume that $\sum_{j=1}^K w[\pi_j] = 1$. Let $\tilde{v}_{1,j} = w[\pi_j]^\eta$ for every $j \in [K]$ be the initial path weights in the PBWM *algorithm. Then, the weighted regret of* PBWM *is bounded as follows:*

$$\mathrm{Reg}_T(\mathrm{PBWM}, \mathcal{C}) \leq \frac{\eta T}{8} + \frac{1}{\eta} \log \left(\sum_{j=1}^K w[\pi_j]^\eta |\mathcal{C} \cap \mathcal{S}_T|^\eta \right),$$

where the upper bound is at most:

$$\frac{\eta T}{8} + \frac{1}{\eta} \log(|\mathcal{C} \cap \mathcal{S}_T|).$$

If we additionally assume that w is supported on more than a single path (i.e. $|\operatorname{supp}(w)| > 1$), then there exists $\eta^* > 0$ (decreasing as a function of T) such that:

$$\mathrm{Reg}_T(\mathrm{PBWM}, \mathcal{C}) \leq \sqrt{\frac{T H_{\eta^*}(w)}{8}} - H_{\eta^*}(w) + \log(K).$$

where $H_\eta(w) = \frac{1}{1-\eta} \log \left(\sum_{j=1}^K w[\pi_j]^\eta \right)$ is the η-Rényi entropy.

For the weighted regret, note that since the η-Rényi entropy is non-negative and increasing in η [Van Erven and Harremos, 2014], the latter bound is at most

$$\sqrt{\frac{T H_0(w)}{8}} + \log(K) = \sqrt{\frac{T \log(|\operatorname{supp}(w)|)}{8}} + \log(K).$$

Moreover, if T is sufficiently large such that $\eta^* < 1$ (e.g. $T \geq \log(|\operatorname{supp}(w)|)$, based on the proof of the theorem), and our path weights w are concentrated on only a subset of the paths $A \subset [K]$ such that

$$\frac{\sum_{j \in A^C} w[\pi_j]}{\sum_{j \in A} w[\pi_j]} < \epsilon(1 - \eta^*),$$

then

$$\begin{aligned} H_\eta^*(w) &= \frac{1}{1-\eta^*} \log\left(\sum_{j=1}^{K} w[\pi_j]^{\eta^*}\right) = \frac{1}{1-\eta^*} \log\left(\sum_{j \in A} w[\pi_j]^{\eta^*} + \sum_{j \in A^c} w[\pi_j]^{\eta^*}\right) \\ &\leq \frac{1}{1-\eta^*} \left[\log\left(\sum_{j \in A} w[\pi_j]^{\eta^*}\right) + \frac{\sum_{j \in A^c} w[\pi_j]^{\eta^*}}{\sum_{j \in A} w[\pi_j]^{\eta^*}}\right] \\ &\leq \frac{1}{1-\eta^*} \log\left(\sum_{j \in A} w[\pi_j]^{\eta^*}\right) + \epsilon \leq \frac{1}{1-\eta^*} \log(|A|) + \epsilon. \end{aligned}$$

Thus, the latter bound can be substantially tighter than the bound in terms of $\log(|\mathcal{C} \cap \mathcal{S}_T|)$.

1.4 Automata operations

While PBWM achieves a sublinear regret bound for both the weighted and unweighted regret, it can be computationally prohibitive. This is because the update at each iteration requires a summation over the number of paths in the competitor set, which can be exponential in the number of total rounds T. For instance, the total number of k-shifting experts is $\mathcal{O}\left(\binom{T-1}{k} N(N-1)^k\right)$.

To design computationally efficient algorithms, we will exploit the properties of the competitor class automaton $\mathcal{C}$. Note that any finite set of competitor paths

can be represented by an automaton. But, different automata accepting the same set may lead to vastly different computational costs.

Our algorithms make use of several common operations for weighted automata: *composition*, *shortest-distance*, and *connection*. In particular, we design an incremental version of the shortest-distance algorithm, which we call IncrementalShortestDistance (IncrSD).

1.4.1 Composition/intersection

Composition (or intersection) is an operation that combines two weighted automata into a new weighted automaton. The resulting automaton accepts the set of strings accepted by both input automata. The weight assigned to any string by the resulting composition is the product of the weights assigned by the input automata: $\mathcal{A}_1 \circ \mathcal{A}_2(x) = \mathcal{A}_1(x)\mathcal{A}_2(x)$.

A standard and efficient method for composing two weighted automata is to pair up matching transitions [Mohri, 2009]. States of $\mathcal{A}_1 \circ \mathcal{A}_2$ are identified with pairs of states of $\mathcal{A}_1$ and $\mathcal{A}_2$: $Q \subseteq Q_1 \times Q_2$, as are the set of initial and final states. Transitions are obtained by matching pairs of transitions from each weighted automaton and multiplying their weights following the result:

$$\left(q_1 \xrightarrow{a/w_1} q_1',\ q_2 \xrightarrow{a/w_2} q_2'\right) \quad \Rightarrow \quad (q_1, q_2) \xrightarrow{a/(w_1 w_2)} (q_1', q_2').$$

In general, the space and time complexity of this composition is $\mathcal{O}(|\mathcal{A}_1||\mathcal{A}_2|)$, since in the worst case all transitions in $\mathcal{A}_1$ could be matched with transitions leaving $\mathcal{A}_2$. In our case, this would result in a complexity of $\mathcal{O}(|\mathcal{C}||\mathcal{S}_T|) = \mathcal{O}(|\mathcal{C}|NT)$. However, composition only actually constructs states and transitions that are reach-

able from the initial states $I_1 \times I_2$. More generally, for an efficient implementation (in terms of data structure), the complexity is linear in the size of the output (disregarding data structure preprocessing).

Now consider the incremental cost of creating states corresponding to $q = (\cdot, t+1)$ and the transitions reaching these states given that all states $q = (\cdot, s)$ with $s \leq t$ and their incoming transitions have been already created. The maximum number of transitions in this step is bounded by the total number of states reachable at time t multiplied by the maximum out-degree of any of these states: $|Q_{\mathcal{C},t}| \max_{q \in Q_{\mathcal{C},t}} |E[q]|$, where $Q_{\mathcal{C},t} = \{q \in Q_{\mathcal{C}} \times Q_{\mathcal{S}_T} \colon q = (\cdot, t)\}$. Thus, the total computational complexity of the composition of $\mathcal{C}$ with $\mathcal{S}_T$ is in $\mathcal{O}\big(\sum_{t=1}^{T} |Q_{\mathcal{C},t}| \max_{q \in Q_{\mathcal{C},t}} |E[q]|\big)$, where $|E[q]| \leq N$ and $|Q_{\mathcal{C},t}| \leq |\mathcal{C}|$. In general, we can expect both quantities to be much smaller.

1.4.2 Shortest-distance

The second operation that we need is a single-source 'shortest-distance' computation over the semiring $(\mathbb{R}_+ \cup \{+\infty\}, +, \times, 0, 1)$, that is, we wish to compute for any state q, $\alpha[q]$ defined by:

$$\alpha[q] = \sum_{\pi \in P(I_{\mathcal{A}}, q)} w_{\mathcal{A}}[\pi],$$

where $P(I_{\mathcal{A}}, q)$ is the set of paths from the initial states $I_{\mathcal{A}}$ to q. For any acyclic automaton $\mathcal{A}$, α can be computed in linear time using a general relaxation-based shortest-distance algorithm with a topological queue discipline [Mohri, 2009]. α can also be computed using other algorithms such as the Viterbi algorithm. In our context, this shortest-distance algorithm can be naturally modified to run in an incremental fashion and extended to the case of automata with failure

Algorithm 2: IncrementalShortestDistance(IncrSD).

Algorithm: IncrSD($\mathcal{C} \cap \mathcal{S}_T$, t, $\boldsymbol{\alpha}$)
$\mathcal{C}_T \leftarrow \mathcal{C} \cap \mathcal{S}_T$
for each $q \in Q_{\mathcal{C}_T}$ **with** $q = (\cdot, t)$ **do**
    $\alpha[q] \leftarrow r[q] \leftarrow 0$
if $t = 1$ **then**
    for each $q \in I_{\mathcal{C}_T}$ **do**
        $\alpha[q] \leftarrow r[q] \leftarrow 1$
    $\mathcal{Q} \leftarrow I_{\mathcal{C}_T}$.
else
    $\mathcal{Q} \leftarrow \{q \in Q_{\mathcal{C}_T} : q = (\cdot, t-1)\}$
while $\mathcal{Q} \neq \varnothing$ **do**
    $q \leftarrow \text{Head}(\mathcal{Q})$
    Dequeue($\mathcal{Q}$)
    $\tilde{r} \leftarrow r[q]$
    $r[q] \leftarrow 0$
    for each $e \in E_{\mathcal{C}_T}[q]$ **do**
        if $\alpha[\text{dest}[e]] \neq \alpha[\text{dest}[e]] + (\tilde{r}w[e])$ **then**
            $\alpha[\text{dest}[e]] \leftarrow \alpha[\text{dest}[e]] + (\tilde{r}w[e])$
            $r[\text{dest}[e]] \leftarrow r[\text{dest}[e]] + (\tilde{r}w[e])$
            if $\text{dest}[e] \notin \mathcal{Q}$ **then**
                Enqueue($\mathcal{Q}$, dest[e])
return $\boldsymbol{\alpha}$

transitions (Section 1.8). We provide the pseudocode of this subroutine as Algorithm 2, IncrementalShortestDistance (IncrSD). This will be a key ingredient for our computationally efficient learning algorithm.

1.4.3 Connection

It is possible that some of the states created by composition will be non-accessible or non-coaccessible. A state $q \in Q$ is non-accessible if there is no path from I to q. A state $q \in Q$ is non-coaccessible if there is no path from q to F. Such states are called *useless* because they do not lie on accepting paths. Connection (or trimming) is a linear-time algorithm that removes these non-accessible states [Mohri, 2009].

1.5 Automata-based algorithm

We now have the tools to present Algorithm 3, AutomataWeightedMajority (AWM), an automata-based online learning algorithm. At the start of the algorithm, we compose the competitor class automaton $\mathcal{C}$ with $\mathcal{S}_T$. For computational efficiency, we apply a connection (or trimming) algorithm to prune useless states from the composed automaton. We then perform a shortest-distance computation *backwards* to compute $\beta[q]$ the sum of the weights of all paths from q to the final states. At each time t, we use the current distribution p_t to sample an expert i_t, receive the loss vector $\mathbf{l}_t$ and incur loss $l_t[i_t]$. We update the weights of the edges in $\mathcal{C} \cap \mathcal{S}_T$ whose destination states have form $(\cdot, t)$ using these losses.

We then compute the flow of each edge e, flow$[e]$, that is the sum of the weights of all paths that pass through e. In the semiring $(\mathbb{R}_+, +, \times, 0, 1)$ that we operate in, this flow can be computed as the product of three components: the shortest-distance from the initial state to the source of e, $\alpha[\text{src}[e]]$, the weight of edge e, $w[e]$, and the shortest distance from the destination state of e to the set of final states in $\mathcal{C} \cap \mathcal{S}_T$, $\beta[\text{dest}[e]]$. We then aggregate these flows by label (i.e. experts) and normalize them to get a distribution over the set of experts. Finally, since the weights of the transitions leading to states $(\cdot, t)$ have been modified, we update the shortest distances to their destination states using our incremental shortest-distance algorithm. The computational cost of this algorithm is linear in the number of these transitions. Note that other shortest-distances need not be modified.

Theorem 1.2 guarantees that AWM performs the same update as PBWM and quantifies its computational cost.

Theorem 1.2 (AutomataWeightedMajority Guarantee)**.** *The update made*

Algorithm 3: AUTOMATAWEIGHTEDMAJORITY(AWM).

Algorithm: AWM($\mathcal{C}$)
$\mathcal{C}_T \leftarrow \mathcal{C} \cap \mathcal{S}_T$
$\mathcal{C}_T \leftarrow \textsc{Connect}(\mathcal{C}_T)$
$\boldsymbol{\beta} \leftarrow \textsc{ShortestDistance}((\mathcal{C}_T)^R, F)$
$w_{\mathcal{C}_T}[I_{\mathcal{C}_T}] \leftarrow w_{\mathcal{C}_T}[I_{\mathcal{C}_T}]/\beta[I_{\mathcal{C}_T}]$
$\boldsymbol{\alpha} \leftarrow 0$
$\mathbf{Flow}_0 \leftarrow 0$
for each $e \in E_{\mathcal{C}_T}$ **with** $\mathrm{dest}[e] = (\cdot, 1)$ **do**
  $\mathrm{flow}[e] \leftarrow \beta[\mathrm{dest}[e]]$
  $\mathrm{Flow}_0[\sigma[e]] \leftarrow \mathrm{Flow}_0[\sigma[e]] + \mathrm{flow}[e]$
  $Z_0 \leftarrow Z_0 + \mathrm{flow}[e]$
$\mathsf{p}_1 \leftarrow \mathrm{Flow}_0/Z_0$
for $t \leftarrow 1$ **to** T **do**
  $i_t \leftarrow \textsc{Sample}(\mathsf{p}_t)$
  $\textsc{Play}(i_t)$
  $\textsc{Receive}(\mathbf{l}_t)$
  $\mathbf{Flow}_t \leftarrow 0$
  for each $e \in E_{\mathcal{C}_T}$ **with** $\mathrm{dest}[e] = (\cdot, t)$ **do**
    $w[e] \leftarrow w[e]e^{-\eta l_{t,\sigma[e]}}$
    $\mathrm{flow}[e] \leftarrow \alpha[\mathrm{src}[e]]w[e]\beta[\mathrm{dest}[e]]$
    $\mathrm{Flow}_t[\sigma[e]] \leftarrow \mathrm{Flow}_t[\sigma[e]] + \mathrm{flow}[e]$
    $Z_t \leftarrow Z_t + \mathrm{flow}[e]$.
  $\mathsf{p}_{t+1} \leftarrow \mathbf{Flow}_t/Z_t$
  $\boldsymbol{\alpha} \leftarrow \textsc{IncrSD}(\mathcal{C}_T, t, \boldsymbol{\alpha})$

by AWM, p_{t+1} *at round* t*, coincides with the update made by* PBWM *for unweighted regret when run with a normalized input* $\mathcal{C} \cap \mathcal{S}_T$*. Moreover, the computational complexity of* AWM *at each round* t *is* $\mathcal{O}\Big(|E[Q_{\mathcal{C}_T,t}]|\Big)$ *and the total computational cost is* $\mathcal{O}\Big(\sum_{t=1}^{T} |E[Q_{\mathcal{C}_T,t}]|\Big)$.

If, after normalization, we exponentiated the weight of every edge to the power η, then we would also recover the update as in PBWM for weighted regret.

Proof. If $\mathcal{C} \cap \mathcal{S}_T$ is normalized as input to PBWM, then it coincides with the result of the normalization of $w_{\mathcal{C}_T}$ at the start of AWM. At any time $t \in [T]$, $\mathrm{Flow}_t[a_i]$

is maintained as $\sum_{e:\,\mathrm{dest}[e]=(\cdot,t),\sigma[e]=a_i} \mathrm{flow}(e)$ that is the sum of the weights of all paths taking action $a_i \in \Sigma$ at time t. The weight of path $\pi = e_1 e_2 \dots e_T$ is then:

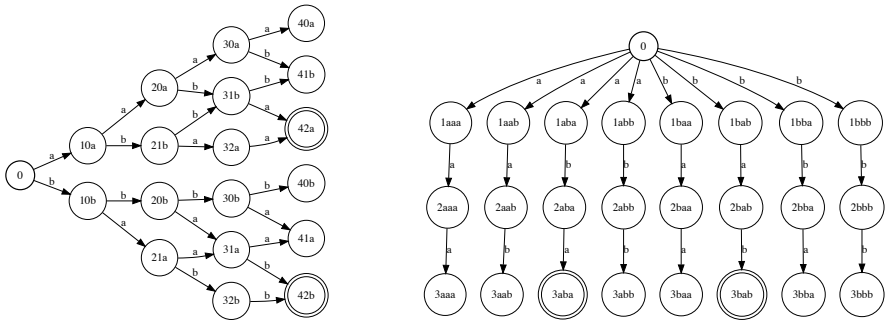
$$w[\pi] \prod_{s=1}^{t} e^{-\eta l_s[\sigma[e_s]]} = w[\pi] e^{-\eta \sum_{s=1}^{t} l_s[\sigma[e_s]]} = \tilde{v}_{t+1,j}.$$

At the start of the algorithm, the normalization and shortest-distance computations take time $\mathcal{O}\left(\sum_{t=1}^{T} |E[Q_{\mathcal{C}_T,t}]|\right)$. Computing the initial probability p_1 takes time $\mathcal{O}(|Q_1|)$. From time $t = 1$ to T, the loss update, flow computation, and flow normalization each take time $\mathcal{O}(|Q_t|)$, and the incremental shortest-distance update takes time $\mathcal{O}\left(|E[Q_{\mathcal{C}_T,t}]|\right)$. Thus, the per-iteration complexity is $\mathcal{O}\left(|E[Q_{\mathcal{C}_T,t}]|\right)$, and the total computational complexity is $\mathcal{O}\left(\sum_{t=1}^{T} |E[Q_{\mathcal{C}_T,t}]|\right)$. □

Notice that the composition of the k-shifting experts competitor class with $\mathcal{C}$ given by Figure 1.2 results in Figure 1.3, so that the per-iteration complexity of AWM for the k-shifting automaton is $\mathcal{O}(N^2 k)$. This is substantially more efficient than the naïve update, which was $\mathcal{O}\left(\binom{T-1}{k} N(N-1)^k\right)$.

In general, the performance of AWM depends not only on the properties of the competitor class automaton but also on those of the resulting automaton after composition with $\mathcal{S}_T$. For instance, we could have also represented the k-shifting experts competitor class using the inefficient representation of Figure 1.3 (b), which would lead to a significantly higher per-iteration computational cost.

It should be noted that AWM can be seen as a generalization of the Expert Hidden Markov Model in [Koolen and de Rooij, 2013] to arbitrary losses. One major difference between their work and ours is the matter of framework and perspective. [Koolen and de Rooij, 2013] assume a Bayesian setting where the prior distribution over expert sequences is given and must be used. In our work, we assume the



(a) (b)

Figure 1.3: (a) An automaton that accepts the composition of k-shifting experts with $\mathcal{S}_T$. (b) An inefficient automaton that accepts the composition of k-shifting experts with $\mathcal{S}_T$.

existence of a competitor automaton, but don't necessarily need to sample from it to make a prediction. This will be crucial in the next section, where we design and use other automata to improve computational efficiency while preserving regret performance.

1.6 Approximation algorithms

The algorithm described in Sections 1.5 was based on exploiting the properties of the competitor class automaton $\mathcal{C}$. AWM was *exact* in that it performed the same computations as the naïve algorithm, PBWM. If the resulting automaton after composition is still large, then the computational cost can still be prohibitive.

In this section, we take a different approach and no longer constrain ourselves to algorithms that perform precisely the same update as PBWM. Instead, we design algorithms to efficiently *approximate* the competitor set $\mathcal{C}_T$. An automaton that is

compact and tightly approximates $\mathcal{C}_T$ can lead to an algorithm that is efficient and achieves a favorable regret.

1.6.1 The effect of automata approximation on regret

We first analyze the approximation cost of such an algorithm. For this, we will need to introduce the notion of ∞-*Rényi divergence* [Rényi et al., 1961]. The ∞-Rényi divergence between two distributions p and q over some set $\mathcal{X}$ is defined to be

$$D_\infty(\mathsf{p}||\mathsf{q}) = \sup_{x \in \mathcal{X}} \log \left[\frac{\mathsf{p}[x]}{\mathsf{q}[x]} \right]. \tag{1.5}$$

Theorem 1.3 (Automata approximation cost)**.** *Assume that $\mathcal{C}_T$ is normalized so that $\sum_{\pi \in \mathcal{C}_T} w_{\mathcal{C}_T}[\pi] = 1$. If* AWM *is run with a weighted automaton $\widehat{\mathcal{C}}$, then its unweighted regret is bounded as follows:*

$$\mathrm{Reg}_T^0(\mathcal{A}, \mathcal{C}) \leq \max_{\pi \in \mathcal{C}_T} \frac{\eta T}{8} + \frac{1}{\eta} \log \left(\frac{1}{w_{\mathcal{C}_T}[\pi]} \right) + \frac{1}{\eta} D_\infty(\mathsf{p}_{\mathcal{C}_T} || \mathsf{p}_{\widehat{\mathcal{C}}_T}),$$

and its weighted regret is bounded as follows:

$$\begin{aligned} \mathrm{Reg}_T(\mathcal{A}, \mathcal{C}) &\leq \frac{\eta T}{8} + \frac{1}{\eta} \log \left(\sum_{\pi \in \widehat{\mathcal{C}}_T} w_{\widehat{\mathcal{C}}_T}[\pi]^\eta |\mathcal{C}_T|^\eta \right) + D_\infty(\mathsf{p}_{\mathcal{C}_T} || \mathsf{p}_{\widehat{\mathcal{C}}_T}) \\ &\leq \frac{\eta T}{8} + \frac{1}{\eta} \log(|\mathcal{C}_T|) + \log \left(\frac{|\mathcal{C}_T|}{|\widehat{\mathcal{C}}_T|} \right). \end{aligned}$$

Proof. The regret guarantee of AWM is based on the fact that it performs the same update as PBWM. By the proof of Theorem 1.1, the unweighted regret of

the algorithm against $\pi \in \mathcal{C}_T$ is bounded by:

$$\sum_{t=1}^{T} \mathsf{p}_t \cdot \mathbf{l}_t - \sum_{t=1}^{T} l_t[z_t] \leq \frac{\eta T}{8} + \frac{1}{\eta} \log\left(\frac{1}{w_{\widehat{\mathcal{C}}_T}[\pi]}\right),$$

where $\sigma[\pi] = z_1^T$.

By comparing the weight of the automaton used in the algorithm, $\widehat{\mathcal{C}}_T$, with the automaton used in the benchmark, $\mathcal{C}_T$, we obtain the bound:

$$\sum_{t=1}^{T} \mathsf{p}_t \cdot \mathbf{l}_t - \sum_{t=1}^{T} l_t[z_t] \leq \frac{\eta T}{8} + \frac{1}{\eta} \log\left(\frac{1}{w_{\mathcal{C}_T}[\pi]}\right) + \frac{1}{\eta} \log\left(\frac{w_{\mathcal{C}_T}[\pi]}{w_{\widehat{\mathcal{C}}_T}[\pi]}\right).$$

This allows us to bound the unweighted regret by:

$$\mathrm{Reg}_T^0(\mathcal{A}, \mathcal{C}) \leq \max_{\pi \in \mathcal{C}_T} \frac{\eta T}{8} + \frac{1}{\eta} \log\left(\frac{1}{w_{\mathcal{C}_T}[\pi]}\right) + \frac{1}{\eta} \log\left(\frac{w_{\mathcal{C}_T}[\pi]}{w_{\widehat{\mathcal{C}}_T}[\pi]}\right).$$

Similarly, for any $\pi \in \mathcal{C}_T$, the weighted regret of the algorithm against π is bounded by:

$$\sum_{t=1}^{T} \mathsf{p}_t \cdot \mathbf{l}_t - \sum_{t=1}^{T} l_t[z_t] - \log\left(w_{\widehat{\mathcal{C}}_T}[\pi]|\mathcal{C}_T|\right) \leq \frac{\eta T}{8} + \frac{1}{\eta} \log\left(\sum_{\pi \in \widehat{\mathcal{C}}_T} w_{\widehat{\mathcal{C}}_T}[\pi]^\eta |\mathcal{C}_T|^\eta\right),$$

so that

$$\begin{aligned}
&\sum_{t=1}^{T} \mathsf{p}_t \cdot \mathbf{l}_t - \sum_{t=1}^{T} l_t[z_t] - \log\left(w_{\mathcal{C}_T}[\pi]|\mathcal{C}_T|\right) \\
&\quad \leq \frac{\eta T}{8} + \frac{1}{\eta} \log\left(\sum_{\pi \in \widehat{\mathcal{C}}_T} w_{\widehat{\mathcal{C}}_T}[\pi]^\eta |\mathcal{C}_T|^\eta\right) + \log\left(\frac{w_{\mathcal{C}_T}[\pi]}{w_{\widehat{\mathcal{C}}_T}[\pi]}\right).
\end{aligned}$$

This then implies that:

$$\mathrm{Reg}_T(\mathcal{A}, \mathcal{C}) \leq \frac{\eta T}{8} + \frac{1}{\eta} \log \left(\sum_{\pi \in \widehat{\mathcal{C}}_T} w_{\widehat{\mathcal{C}}_T}[\pi]^\eta |\mathcal{C}_T|^\eta \right) + \log \left(\sup_{\pi \in \mathcal{C}_T} \frac{w_{\mathcal{C}_T}[\pi]}{w_{\widehat{\mathcal{C}}_T}[\pi]} \right).$$

Furthermore, as in the proof of Theorem 1.1, we can further bound the above quantity by:

$$\frac{\eta T}{8} + \frac{1}{\eta} \log \left(\sum_{\pi \in \widehat{\mathcal{C}}_T} \frac{|\mathcal{C}_T|^\eta}{|\widehat{\mathcal{C}}_T|^\eta} \right) + \log \left(\sup_{\pi \in \mathcal{C}_T} \frac{w_{\mathcal{C}_T}[\pi]}{w_{\widehat{\mathcal{C}}_T}[\pi]} \right).$$

□

Theorem 1.3 quantifies the extra cost of using a surrogate automaton during learning as $\frac{1}{\eta} D_\infty(\mathsf{p}_{\mathcal{C}_T} || \mathsf{p}_{\widehat{\mathcal{C}}_T})$ for unweighted regret and as $D_\infty(\mathsf{p}_{\mathcal{C}_T} || \mathsf{p}_{\widehat{\mathcal{C}}_T})$ for weighted regret. This is tight, since it is always possible that the best path in hindsight in the regret definition may also be the one maximizing the log-ratio.

This suggests that using an approximate automaton $\widehat{\mathcal{C}}$ for which the size of $\widehat{\mathcal{C}}_T$ is favorable and $D_\infty(w_{\mathcal{C}_T} || w_{\widehat{\mathcal{C}}_T})$ is small would both lead to an efficient algorithm and a favorable regret guarantee.

A consequence of this bound is a general algorithm. Given a family of automata $\mathcal{C}$ with a relatively small number of states and transitions, the algorithm can be formulated as solving the problem:

$$\min_{\widehat{\mathcal{C}}_T \in \mathcal{C}} D_\infty(\mathsf{p}_{\mathcal{C}_T} || \mathsf{p}_{\widehat{\mathcal{C}}_T}). \tag{1.6}$$

When the set of distributions associated to $\mathcal{C}$ is convex, this is a convex optimization problem, since $\mathsf{q} \mapsto \log(\mathsf{p}/\mathsf{q})$ is a convex function and the supremum of convex

functions is convex. The choice of the set $\mathcal{C}$ is subject to a trade-off: approximation accuracy versus computational efficiency of using elements in $\mathcal{C}$. This raises a model selection question for which we discuss in detail a solution in Section 1.6.5: given a sequence of families $(\mathcal{C}_n)_{n\in\mathbf{N}}$ with growing complexity and computational cost, the problem consists of selecting the best n.

For general choices of $\mathcal{C}$, this problem can be complex and relates to other standard automata learning problems [Pitt and Warmuth, 1993, Balle and Mohri, 2012, Hsu et al., 2012, Balle and Mohri, 2015]. In the following, we will consider the case where the family $\mathcal{C}$ of weighted automata is that of *n-gram models*, for which we can upper bound the computational complexity.

1.6.2 n-gram models

An n-gram language model is a Markovian model of order $(n-1)$ defined over Σ^*. By the chain rule of probability, $\mathsf{p}_{\mathcal{A}}[x] = \prod_{t=1}^{T} \mathsf{p}_{\mathcal{A}}[x_t|x_1^{t-1}]$. For an n-gram model, conditioning is done only on a history of length $(n-1)$, thus $\mathsf{p}_{\mathcal{A}}[x] = \prod_{t=1}^{T} \mathsf{p}[x_t|x_{t-n+1}^{t-1}]$. An n-gram model can be compactly represented by a weighted automaton with states $Q = \Sigma^{n-1}$ encoding each possible history, and with one transition corresponding to each n-gram $x_1 \cdots x_n \in \Sigma^n$ from state x_1^{n-1} to state x_2^n and labeled with x_n. The natural automata representation for n-gram models are *stochastic* automata, where the weights of the outgoing transitions for any state are non-negative and sum up to one. Any probabilistic weighted automaton can be converted into a stochastic automaton using the *weight-pushing* algorithm [Mohri, 2009], which takes linear time and consists of a shortest-distance computation combined with a reweighting of the transition weights, initial weights, and final weights in a way that *pushes* the weights towards the initial states.

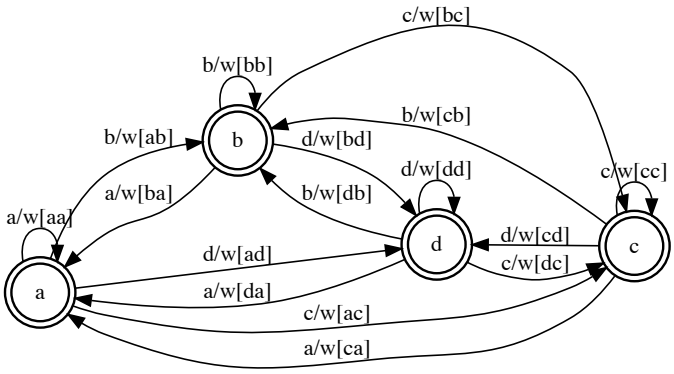


Figure 1.4: A bigram language model over the alphabet $\Sigma = \{a, b, c, d\}$.

Each state in an n-gram model encodes a particular history, so that for stochastic automata, the outgoing transition weights are the conditional probabilities given that history. Since these probabilities define an n-gram model, an n-gram model can be interpreted as a product of simplices.

See Figure 1.4 for an example of an n-gram language model. Language models of this form are commonly used in language and speech processing [Jelinek and Mercer, 1980, Katz, 1987, Ney et al., 1994, Chen and Goodman, 1998, Allauzen et al., 2003]. In these applications, the models are typically *smoothed* since they are trained on a finite sample. In our case, smoothing will not be needed since we can train directly on $\mathcal{C}_T$, which we interpret as the full language.

One key advantage of n-gram models in this context is that the per-iteration complexity can be bounded in terms of the number of symbols. Since an n-gram model has at most $|\Sigma|^{n-1}$ states, its per-iteration computational cost is in $\mathcal{O}\big(|\Sigma|^n\big)$ as each state can take one of $|\Sigma|$ possible transitions. For n small, this can be very advantageous compared to the original $\mathcal{C}_T$, since the maximum out-degree of states

reached by strings of length t in the latter may be very large.

In what follows, we denote by $\mathcal{P}_n$ the family of n-gram language models. Since $\mathcal{P}_n$ can be written as the product of simplices, it is therefore a convex set. If we denote by $\mathsf{p}_{\mathcal{C}_T}$ the target distribution of sequences, then we can apply well-known techniques to learn a language model over the language $\mathcal{C}_T$.

1.6.3 Maximum likelihood n-gram models

A standard method for learning n-gram models is via maximum likelihood, which is equivalent to minimizing the relatively entropy between the target distribution $\mathsf{p}_{\mathcal{C}_T}$ and the language model. This is defined by the optimization problem:

$$\min_{\mathsf{p}_{\mathcal{A}_n} \in \mathcal{P}_n} D(\mathsf{p}_{\mathcal{C}_T} || \mathsf{p}_{\mathcal{A}_n}), \tag{1.7}$$

where, for any two distributions p and q, $D(\mathsf{p}||\mathsf{q}) = \sum_x \mathsf{p}[x] \log\left(\frac{\mathsf{p}[x]}{\mathsf{q}[x]}\right)$. Observe that, in general, the solution of this optimization problem is not guaranteed to provide an upper bound on the ∞-Rényi divergence since the ∞-Rényi divergence is an upper bound on the relative entropy[2]. However, as we shall see later (e.g., Theorem 1.4), in some cases, maximum likelihood solutions do benefit from favorable guarantees.

Maximum likelihood n-gram solutions in particular are simple. For standard text, the weight of each transition is the frequency of appearance of the corresponding n-gram in the text. For a probabilistic $\mathcal{C}_T$, the weight can be similarly obtained from the expected count of the n-gram in the paths of $\mathcal{C}_T$, where the expectation is taken over the probability distribution defined by $\mathcal{C}_T$ and can be computed efficiently [Allauzen et al., 2003].

[2]The relative entropy also coincides with the 1-Rényi divergence

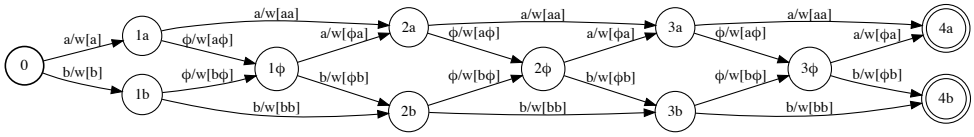


Figure 1.5: An illustration of $\mathcal{A}_2 \cap \mathcal{S}_T$, a bigram model approximating the k-shifting automaton. Note that φ-Convert has been applied to the bigram, making it smaller than a standard bigram model.

Maximum likelihood n-gram models can further benefit from φ-conversion using the algorithms presented in Section 1.8. This can reduce the size of $\mathcal{A}_n$ and improve its computational efficiency without affecting its accuracy.

As an example, we can compute the bigram approximation to the k-shifting automaton. Remarkably, this will coincide with the Fixed-Share algorithm of Herbster and Warmuth [1998]. Thus, we can view and motivate the design of Fixed-Share as a bigram approximation of the desired competitor automaton, that is the family of k-shifting sequences.

Theorem 1.4 (Bigram approximation of k-shifting automaton)**.** *Let $\mathcal{C}_T$ be the k-shifting automaton for some k. Then, the bigram model $\mathcal{A}_2$ obtained using relative entropy satisfies:*

$$\mathsf{p}_{\mathcal{A}_2}[a_1 a_2] = \frac{1}{N}\Big(1 - \frac{k}{(T-1)}\Big)\, 1_{a_1 = a_2} + \frac{k}{(T-1)(N-1)N}\, 1_{a_1 \neq a_2}.$$

Moreover, the approximation error is bounded as follows:

$$\sup_{\pi \in \mathcal{C}_T} \log\left(\frac{\mathsf{p}_{\mathcal{C}_T}[\pi]}{\mathsf{p}_{\mathcal{A}_2}[\pi]}\right) \leq -\log\left(1 - 2e^{-\frac{1}{12k}}\right).$$

Proof. Let $a_1, a_2 \in \Sigma$. Then

$$\mathsf{p}_{\mathcal{A}_2}[a_2|a_1] = \mathsf{p}_{\mathcal{A}_2}[a_2|a_1, a_2 = a_1]\,\mathsf{p}_{\mathcal{A}_2}[a_2 = a_1] + \mathsf{p}_{\mathcal{A}_2}[a_2|a_1, a_2 \neq a_1]\,\mathsf{p}_{\mathcal{A}_2}[a_2 \neq a_1]$$

Consider first the case where $a_2 = a_1$. Then $\mathsf{p}_{\mathcal{A}_2}[a_2|a_1, a_2 = a_1] = 1$, and $\mathsf{p}_{\mathcal{A}_2}[a_2 = a_1]$ is the expected number of times that we see label a_2 agreeing with label a_1. Since $\mathsf{p}_{\mathcal{C}_T}$ is uniform for the k-shifting automaton, the expected counts are pure counts, and the probability that we see two consecutive labels agreeing is $1 - \frac{k}{T-1}$. Now, consider the case where $a_2 \neq a_1$. By symmetry, $\mathsf{p}_{\mathcal{A}_2}[a_2|a_1, a_2 \neq a_1] = \frac{1}{N-1}$, since a_2 is equally likely to be any of the other $N-1$ labels. Moreover, $\mathsf{p}_{\mathcal{A}_2}[a_2 \neq a_1] = \frac{k}{T-1}$.

Thus, the following holds:

$$\mathsf{p}_{\mathcal{A}_2}[a_2|a_1] = \frac{1}{N-1}\frac{k}{T-1}1_{a_1 \neq a_2} + \left(1 - \frac{k}{T-1}\right)1_{a_1 = a_2}.$$

Moreover, by symmetry, we can write $\mathsf{p}_{\mathcal{A}_2}[a_1] = \frac{1}{N}$, and therefore,

$$\mathsf{p}_{\mathcal{A}_2}[a_1 a_2] = \mathsf{p}_{\mathcal{A}_2}[a_2|a_1]\mathsf{p}_{\mathcal{A}_2}[a_1] = \frac{k}{N(N-1)(T-1)}1_{a_1 \neq a_2} + \left(\frac{T-1-k}{N(T-1)}\right)1_{a_1 = a_2}.$$

To bound the approximation error in the regret, notice that for any string x accepted

by $\mathcal{C}_T$:

$$
\begin{aligned}
&\log\left(\frac{\mathsf{p}_{\mathcal{C}_T}[x]}{\mathsf{p}_{\mathcal{A}_2}[x]}\right) \\
&= \log\left(\frac{1}{\mathsf{p}_{\mathcal{A}_2}[x]}\right) - \log(|\mathcal{C}_T|) \\
&\quad \text{(since the } k\text{-shifting automaton has uniform weights)} \\
&= \log\left(\frac{1}{\mathsf{p}_{\mathcal{A}_2}[z = x | z \in \mathcal{C}_T]\mathsf{p}_{\mathcal{A}_2}[z \in \mathcal{C}_T] + \mathsf{p}_{\mathcal{A}_2}[z = x | z \notin \mathcal{C}_T]\mathsf{p}_{\mathcal{A}_2}[z \notin \mathcal{C}_T]}\right) \\
&\quad - \log(|\mathcal{C}_T|) \\
&= \log\left(\frac{1}{\frac{1}{|\mathcal{C}_T|}\mathsf{p}_{\mathcal{A}_2}[z \in \mathcal{C}_T] + \mathsf{p}_{\mathcal{A}_2}[z = x | z \notin \mathcal{C}_T]\mathsf{p}_{\mathcal{A}_2}[z \notin \mathcal{C}_T]}\right) - \log(|\mathcal{C}_T|) \\
&\quad \text{(since } \mathsf{p}_{\mathcal{A}_2} \text{ is uniform on } \mathcal{C}_T) \\
&\leq \log\left(\frac{|\mathcal{C}_T|}{\mathsf{p}_{\mathcal{A}_2}[z \in \mathcal{C}_T]}\right) - \log(|\mathcal{C}_T|) \\
&= \log\left(\frac{1}{\mathsf{p}_{\mathcal{A}_2}[z \in \mathcal{C}_T]}\right).
\end{aligned}
$$

The probability that a string z is accepted by $\mathcal{C}_T$ (under the distribution $\mathsf{p}_{\mathcal{A}_2}$) is equal to the probability that it admits exactly k shifts. Let $\xi_t = 1_{\{z \text{ shifts from } t-1 \text{ to } t\}}$ be a random variable indicating whether there is a shift at the t-th symbol in sequence z. This is a Bernoulli random variable bounded by 1 with mean $\frac{k}{T-1}$ and variance $\frac{k}{T-1}(1 - \frac{k}{T-1})$. Since each shift occurs with probability $\frac{k}{T-1}$, we can use Sanov's theorem [Sanov, 1957] to write the following bound:

$$
\mathsf{p}_{\mathcal{A}_2}[z \notin \mathcal{C}_T] = \mathsf{p}_{\mathcal{A}_2}\left[\left|\sum_{t=2}^{T} \xi_t - k\right| > \frac{1}{2}\right] \leq 2e^{-(T-1)u},
$$

where

$$u = (T-1)\min\left\{D\left(\frac{k+(1/2)}{T-1}||\frac{k}{T-1}\right), D\left(\frac{k-(1/2)}{T-1}||\frac{k}{T-1}\right)\right\},$$

and where $D(p||q)$ is the binary entropy.

We will now use the following technical lemma, which we prove in Appendix A.1.

Lemma 1.1 (Binary entropy bound)**.** *The following inequalities hold for the binary entropy:*

$$-D\left(\frac{k+(1/2)}{T-1}||\frac{k}{T-1}\right) \leq -\frac{1}{12k(T-1)}$$

$$-D\left(\left(1-\frac{1}{2k}\right)\frac{k}{T-1}||\frac{k}{T-1}\right) \leq -\frac{\frac{1}{4k^2}\frac{k}{T-1}}{2} = -\frac{1}{8k(T-1)}.$$

Using this result, we can further bound the approximation error in the regret bound by:

$$\log\left(\frac{1}{\mathsf{p}_{\mathcal{A}_2}[z \in \mathcal{C}_T]}\right) \leq \log\left(\frac{1}{1-2e^{-\frac{1}{12k}}}\right) = -\log\left(1-2e^{-\frac{1}{12k}}\right).$$

□

To the best of our knowledge, this is the first framework that motivates the design of Fixed-Share with a focus on minimizing tracking regret. Other works that have recovered Fixed-Share (e.g. [Cesa-Bianchi et al., 2012, Koolen and de Rooij, 2013, György and Szepesvári, 2016]) have generally viewed the algorithm itself as the main focus.

As already pointed out, the bigram approximation of the k-shifting automaton has a per-iteration computational cost of $\mathcal{O}(|\Sigma|^2)$ transitions. At each time t, every state of the form $(\cdot, t)$ corresponds to one of the labels in Σ, and admits a transition

to a state in $(\cdot, t+1)$ corresponding to the same label as well as $|\Sigma| - 1$ transitions to states corresponding to other labels. In Section 1.8, we will introduce the notion of φ-transition and φ-automata to design algorithms that can reduce the size of an automaton while preserving its language and weights. Using a new algorithm φ-Convert on $\mathcal{A}_2 \cap \mathcal{S}_T$, we will significantly reduce the number of transitions in the bigram model so that there are only $\mathcal{O}(|\Sigma|)$ states and transitions reachable at every time. The computational cost of using φ-AWM, another new algorithm, with this new φ-automaton coincides with the one described originally in [Herbster and Warmuth, 1998]. See Figure 1.5 for an illustration.

Our derivation of Fixed-Share also allows us to naturally generalize the setting of standard k-shifting experts to k-shifting experts with non-uniform weights. Specifically, consider the case where $\mathcal{C}_T$ is an automaton accepting up to k-shifts but where the shifts now occur with probability $\mathsf{p}_{\mathcal{C}_T}[a_2|a_1, a_1 \neq a_2] \neq \frac{1}{N-1} 1_{\{a_2 \neq a_1\}}$. Since the bigram approximation will remain exact on $\mathcal{C}_T$, we recover the exact same guarantee as in Theorem 1.4.

The proof technique of Theorem 1.4 is illustrative because it reveals that the maximum likelihood n-gram model has low approximation error whenever (1) the model's distribution is proportional to the distribution of $\mathcal{C}_T$ on $\mathcal{C}_T$'s support and (2) most of the model's mass lies on the support of $\mathcal{C}_T$. When the automaton $\mathcal{C}_T$ has uniform weights, then condition (1) is satisfied when the n-gram model is uniform on $\mathcal{C}_T$. This is true whenever all sequences in $\mathcal{C}_T$ have the same set of n-gram counts, and every permutation of symbols over these counts is a sequence that lies in $\mathcal{C}_T$, which is the case for the k-shifting automaton. Condition (2) is satisfied when n is large enough, which necessarily exists since the distribution is exact for $n = T$. On the other hand, note that a unigram approximation would

Algorithm 4: PROD-EG.

Algorithm: PROD-EG($\mathsf{p}_1 \in (\Delta_N)^m$, η)

for $s = 1, 2, \ldots, \tau$ **do**
  PLAY(p_s)
  RECEIVE($\nabla f(\mathsf{p}_s)$)
  for $j = 1, 2, \ldots, m$ **do**
    for $i = 1, 2, \ldots, N$ **do**
      $\mathsf{p}_{s+1,j}[i] = \mathsf{p}_{s,j} e^{-\eta \frac{\partial f}{\partial \mathsf{p}_j[i]}(\mathsf{p}_{s,j})}$

have satisfied condition (1) but not condition (2) for the k-shifting automaton.

1.6.4 Minimum Rényi divergence n-gram models

Unlike maximum likelihood n-gram models, which in general do not benefit from an approximation guarantee, in this section we discuss minimum Rényi divergence (or ∞-Rényi divergence) n-gram models, which are directly obtained by solving Problem 1.6 over the family of n-grams models. This corresponds to the following optimization problem:

$$\min_{\mathsf{p}_{\mathcal{A}_n} \in \mathcal{P}_n} D_\infty(\mathsf{p}_{\mathcal{C}_T} || \mathsf{p}_{\mathcal{A}_n}) = \min_{\mathsf{p}_{\mathcal{A}_n} \in \mathcal{P}_n} \sup_{x \in \mathcal{C}_T} \log \left(\frac{\mathsf{p}_{\mathcal{C}_T}[x]}{\mathsf{p}_{\mathcal{A}_n}[x]} \right). \tag{1.8}$$

This is a convex optimization problem over a product of $\sum_{j=1}^{n} |\Sigma|^{n-j}$ simplices. The problem can be solved using as an optimization algorithm an extension of the Exponentiated Gradient algorithm developed by Kivinen and Warmuth [1997], which we call PROD-EG. The pseudocode of the algorithm, which is based on a simple multiplicative update, is given in Algorithm 4. The following provides a general guarantee for the convergence of the algorithm. Its proof is provided in Appendix A.1.

Theorem 1.5 (PRODUCT-EXPONENTIATED GRADIENT (PROD-EG))**.**

Let $(\Delta_N)^m$ be the product of m $(N-1)$-dimensional simplices, and let $f\colon (\Delta_N)^m \to \mathbb{R}$ be a loss function whose partial subgradients have absolute values all bounded by L. Let $\mathsf{p}_{1,j}[i] = \frac{1}{N}$ for $i \in [N]$ and $j \in [m]$. Then, Prod-EG *benefits from the following guarantee:*

$$f\left(\frac{1}{\tau}\sum_{s=1}^{\tau}\mathbf{p}_s\right) - f(\mathbf{p}^*) \leq \frac{1}{\eta\tau}m\log(N) + \eta 2L.$$

For the minimum Rényi divergence optimization problem (1.8), we can apply Prod-EG to the product of $m = \sum_{j=1}^{n}|\Sigma|^{n-j}$ simplices, each one corresponding to a conditional probability with a specific history. First, we remark that the subgradient of the maximum of a family of convex functions at a point can always be chosen from the subgradient of the maximizing function at that point. Specifically, let $\{f_\alpha\}_{\alpha\in\mathcal{A}}$ be a family of convex functions, and let $\alpha(x) = \operatorname{argmax}_\alpha f_\alpha(x)$. Then, it follows that

$$\max_\alpha f_\alpha(x) - \max_\alpha f_\alpha(y) \geq f_{\alpha(y)}(x) - f_{\alpha(y)}(y) \geq \langle \nabla f_{\alpha(y)}(y), x - y\rangle.$$

Let π be the maximizing path of the minimum Rényi divergence objective. We will use the $\vee$ symbol to denote the maximum between two values, and the $\wedge$ symbol

the denote the minimum. We can then write

$$\begin{aligned}
\log\left(\frac{\mathsf{p}_{\mathcal{C}_T}[\pi]}{\mathsf{p}[\pi]}\right) &= \log\left(\mathsf{p}_{\mathcal{C}_T}[\pi]\right) + \sum_{t=1}^{T} \log\left(\frac{1}{\mathsf{p}(\pi_t|\pi_{t-n+1\vee 1}^{t-1})}\right) \\
&= \log\left(\mathsf{p}_{\mathcal{C}_T}[\pi]\right) + \sum_{t=1}^{T} \sum_{z_1^{n\wedge t}\in\Sigma^{n\wedge t}} 1_{\pi_{t-n+1\vee 1}^{t}=z_1^{n\wedge t}} \log\left(\frac{1}{\mathsf{p}(\pi_t|\pi_{t-n+1\vee 1}^{t-1})}\right) \\
&= \log\left(\mathsf{p}_{\mathcal{C}_T}[\pi]\right) + \sum_{t=1}^{T} \sum_{z_1^{n\wedge t}\in\Sigma^{n\wedge t}} 1_{\pi_{t-n+1\vee 1}^{t}=z_1^{n\wedge t}} \log\left(\frac{1}{\mathsf{p}(z_{n\wedge t}|z_1^{(n\wedge t)-1})}\right) \\
&= \log\left(\mathsf{p}_{\mathcal{C}_T}[\pi]\right) + \sum_{t=1}^{T} \sum_{j=1}^{n} 1_{j=n\wedge t} \sum_{z_1^{j}\in\Sigma^{j}} 1_{\pi_{t-n+1\vee 1}^{t}=z_1^{j}} \log\left(\frac{1}{\mathsf{p}(z_j|z_1^{j-1})}\right) \\
&= \log\left(\mathsf{p}_{\mathcal{C}_T}[\pi]\right) + \sum_{j=1}^{n} \sum_{z_1^{j}\in\Sigma^{j}} \sum_{t=1}^{T} 1_{j=n\wedge t} 1_{\pi_{t-n+1\vee 1}^{t}=z_1^{j}} \log\left(\frac{1}{\mathsf{p}(z_j|z_1^{j-1})}\right),
\end{aligned}$$

so that its partial derivative with respect to $\mathsf{p}(z_j|z_1^{j-1})$ is:

$$\frac{\partial}{\partial \mathsf{p}(z_j|z_1^{j-1})} \log\left(\frac{\mathsf{p}_{\mathcal{C}_T}[\pi]}{\mathsf{p}[\pi]}\right) = \sum_{t=1}^{T} 1_{j=n\wedge t} 1_{\pi_{t-n+1\vee 1}^{t}=z_1^{j}} \frac{-1}{\mathsf{p}(z_j|z_1^{j-1})}.$$

Thus, by tuning Prod-EG with an adaptive learning rate

$$\eta_t \propto \frac{1}{\sqrt{\sum_{s=1}^{t} \left\| \nabla \log\left(\frac{\mathsf{p}_{\mathcal{C}_T}[\pi(s)]}{\mathsf{p}_s[\pi(s)]}\right)\right\|_\infty^2}},$$

where $\pi(s) = \operatorname{argmax}_{\pi\in\mathcal{C}_T} \log\left(\frac{\mathsf{p}_{\mathcal{C}_T}[\pi]}{\mathsf{p}_s[\pi]}\right)$, we can derive the following guarantee for Prod-EG applied to the n-gram approximation problem.

Corollary 1.1 (n-gram approximation guarantee)**.** *There exists an optimization algorithm outputting a sequence of conditional probabilities* $(\mathsf{p}_s)_{s=1}^{\infty}$ *such that* $\left(\frac{1}{\tau}\sum_{s=1}^{\tau} \mathsf{p}_s\right)$ *approximates the* ∞*-Rényi optimal* n*-gram solution with the following*

guarantee:

$$F\left(\frac{1}{\tau}\sum_{s=1}^{\tau}\mathsf{p}_s\right) - F(\mathsf{p}^*)$$
$$\leq \sqrt{\frac{2N^n \log(N) \sum_{s=1}^{\tau} \max_{\substack{j\in\{1,\ldots,n\}\\ z_1^j\in\Sigma^j}} \left| \sum_{t=1}^{T} 1_{j=n\wedge t} 1_{\pi_{t-n+1\vee 1}^t(s)=z_1^j} \frac{1}{\mathsf{p}_s(z_j|z_1^{j-1})}\right|}{(N-1)T^2}}.$$

Each iteration of PROD-EG admits a computational complexity that is linear in the dimension of the feature space. Since we have specified an n-gram model as the product of $\frac{N^n-1}{N-1}$ simplices, the total per-iteration cost of solving the convex optimization problem is in $\mathcal{O}\left(\frac{N(N^n-1)}{N-1}\right) = \mathcal{O}(N^n)$. Since the minimum Rényi divergence is not Lipschitz, the maximizing ratio in the convergence guarantee may also become large when the choice of n is too small. In all cases, observe that this approximation problem can be solved offline.

If we restrict ourselves to $\mathcal{C}_T$ with uniform weights and $|\Sigma| = 2$, then we can also provide an explicit solution for unigram automata. The solution is obtained from the paths with the smallest number of occurrences of each symbol, which can be straightforwardly found via a shortest-path algorithm in linear time.

Theorem 1.6 (∞-Rényi divergence unigram solution for uniform weight competitors and $|\Sigma| = 2$)**.** *Assume that $\mathcal{C}_T$ admits uniform weights over all paths and $|\Sigma| = 2$. For $j \in \{1, 2\}$, let $n(a_j)$ be the smallest number of occurrences of a_j in a path in $\mathcal{C}_T$. For $j \neq k$, define*

$$\mathsf{p}_k(a_j) = \frac{\max\left\{1, \frac{n(a_j)}{T-n(a_j)}\right\}}{1 + \max\left\{1, \frac{n(a_j)}{T-n(a_j)}\right\}}.$$

Let k^* be the solution of the following optimization problem:

$$\max_{k\in\{1,2\}} \max_{j\in\{1,2\}\setminus\{k\}} n(a_j)\log \mathsf{p}_k(a_j) + [T - n(a_j)]\log\left(1 - \mathsf{p}_k(a_j)\right).$$

Then, p_{k^*} is the solution of the unigram ∞-Rényi divergence optimization problem.

Proof. We seek a unigram distribution $\mathsf{p}_{\mathcal{A}_{1,T}}$ that is a solution of:

$$\min_{\mathsf{p}_{\mathcal{A}_1}\in\mathcal{P}_1} \sup_{\pi\in\mathcal{C}_T} \log\left(\frac{\mathsf{p}_{\mathcal{C}_T}[\pi]}{\mathsf{p}_{\mathcal{A}_1}[\pi]}\right).$$

Since $\mathcal{C}_T$ admits uniform weights, $\mathsf{p}_{\mathcal{C}_T}[\pi] = \frac{1}{|\mathcal{C}_T|}$, and since $\mathcal{A}_{1,T}$ is a unigram automaton, $\log \mathsf{p}_{\mathcal{A}_1}[\pi]$ can be expressed as follows:

$$\log \mathsf{p}_{\mathcal{A}_1}[\pi] = n_\pi(a_1)\log p(a_1) + [T - n_\pi(a_1)]\log\left(1 - p(a_1)\right),$$

where $p(a_j)$ is the automaton's weight on transitions labeled with a_j and $n_\pi(a_j)$ is the count of a_j in path π. Thus, the optimization problem is equivalent to the following problem:

$$-\max_{p(a_1)\in[0,1]} \min_{\pi\in\mathcal{C}_T} n_\pi(a_1)\log p(a_1) + [T - n_\pi(a_1)]\log\left(1 - p(a_1)\right).$$

Denote the objective by $F(p(a_1), n_\pi(a_1))$. Then, the partial derivatives with respect to the label counts are given by

$$\frac{\partial F}{\partial n_\pi(a_1)} = \log p(a_1) - \log\Big(1 - p(a_1)\Big).$$

Thus, $\frac{\partial F}{\partial n_\pi(a_1)} \geq 0$ if and only if $\mathsf{p}(a_1) \geq 1 - \mathsf{p}(a_1)$. Furthermore, if $\mathsf{p}(a_1) \geq 1 - \mathsf{p}(a_1)$, then the path π chosen in the optimization problem is the path with the minimal

count of symbol a_1. Similarly, if $\mathsf{p}(a_2) \geq 1 - \mathsf{p}(a_2)$, then the path π chosen in the optimization problem is the path with minimal count of a_2.

Since we have either $\mathsf{p}(a_1) \geq \mathsf{p}(a_2)$ or vice versa (potentially both), we can write the optimization problem as:

$$- \max_{k \in \{1,2\}} \max_{\substack{p(a_j) \geq 1 - \mathsf{p}(a_j) \\ j \neq k}} \min_{\{n_\pi(a_j)\}_{j \neq k} \colon \pi \in \mathcal{C}_T} n_\pi(a_j) \log \mathsf{p}(a_j) + [T - n_\pi(a_j)] \log (1 - \mathsf{p}(a_j)) .$$

Given $k \in \{1, 2\}$, let $\pi(k)$ be the path that minimizes $n_\pi(a_j)$ over all π for $j \neq k$. Denote these counts $n_{\pi(k)}(a_j)$ by $n(a_j)$. Then we can rewrite the objective as:

$$- \max_{k=1,2,\ldots,N} \max_{\substack{\mathsf{p}(a_j) \geq 1 - \mathsf{p}(a_j) \\ j \neq k}} n(a_j) \log \mathsf{p}(a_j) + [T - n(a_j)] \log (1 - \mathsf{p}(a_j)) .$$

Denote the objective for this new term by $\tilde{F}_k$, which is a function of $\mathsf{p}(a_j)$. The partial derivative of $\tilde{F}_k$ with respect to $\mathsf{p}(a_j)$ is:

$$\frac{\partial \tilde{F}_k}{\partial \mathsf{p}(a_j)} = \frac{n(a_j)}{p(a_j)} - \frac{T - n(a_j)}{1 - \mathsf{p}(a_j)},$$

which is equal to 0 if and only if

$$\mathsf{p}(a_j) = \frac{n(a_j)}{T - n(a_j)} (1 - \mathsf{p}(a_j)) = \max \left\{ 1, \frac{n(a_j)}{T - n(a_j)} \right\} (1 - \mathsf{p}(a_j)) .$$

The last equality follows from our assumption that $\mathsf{p}(a_j) \geq 1 - \mathsf{p}(a_j)$. Now, let $\mathsf{p}_k(a_j)$ denote the probabilities that we have just computed. Then, we can write the optimization problem of $\tilde{F}_k$ as:

$$- \max_{k \in \{1,2\}, j \in \{1,2\} \setminus \{k\}} n(a_j) \log \mathsf{p}_k(a_j) + [T - n(a_j)] \log (1 - \mathsf{p}_k(a_j)) .$$

□

Theorem 1.6 shows that the solutions of the ∞-Rényi divergence optimization are based on the n-gram counts of sequences in $\mathcal{C}_T$ with "high entropy". This can be very different from the maximum likelihood solutions, which are based on the average n-gram counts. For instance, suppose we are under the assumptions of Theorem 1.6, and specifically, assume that there are T sequences in $\mathcal{C}_T$. Assume that one of the sequences has $\left(\frac{1}{2}+\gamma\right)T$ occurrences of a_1 for some small $\gamma > 0$ and that the other $T-1$ sequences have $T-1$ occurrences of a_1. Then, $n(a_1) = \left(\frac{1}{2}+\gamma\right)T$, and the solution of the ∞-Rényi divergence optimization problem is given by $\mathsf{p}_\infty(a_1) = \frac{1+2\gamma}{2}$ and $\mathsf{p}_\infty(a_2) = \frac{1-2\gamma}{2}$. On the other hand, the maximum-likelihood solution would be $\mathsf{p}_1(a_1) = 1 + \frac{\gamma}{T} - \frac{3}{2T} + \frac{1}{T^2} \approx 1$ and $\mathsf{p}_1(a_2) = \frac{3}{2T} - \frac{\gamma}{T} - \frac{1}{T^2} \approx 0$ for large T.

1.6.5 Model selection

The previous section introduced a method to learn a single n-gram model based on the competitor automaton $\mathcal{C}_T$. In practice, one seeks an n-gram model that balances the tradeoff between approximation error and computational cost.

Assume that we have a maximum per-iteration computational budget B. We thus seek the most computationally efficient n-gram model within our budget that does not contribute to an increase in regret. Let $\mathbf{p}_n$ be an n-gram model returned by Prod-EG. By Corollary 1.1, we can write $F(\mathbf{p}_n) - \mathrm{Gap}_{\tau_{\mathrm{opt}},n} \leq F(\mathbf{p}_n^*)$, where $\mathbf{p}_n^*$ is the optimal n-gram model minimizing the objective and $\mathrm{Gap}_{\tau_{\mathrm{opt}},n}$ is the expression quantifying the gap between Prod-EG and the optimal solution after running the algorithm for τ_{opt} iterations. Thus, if $F(\mathbf{p}_n) - \mathrm{Gap}_{\tau_{\mathrm{opt}},n} > \sqrt{T}$ for some n, then

Algorithm 5: n-gramSelect.

Algorithm: n-gramSelect($\mathcal{C}_T$, τ_{opt}, B)

$n \leftarrow 1$
$\mathbf{p}_n \leftarrow \frac{1}{|\Sigma|}$
$s \leftarrow 0$
while $s \leq \tau_{opt}$ **do**
  $\mathbf{p}_n \leftarrow$ Prod-EG-Update($\mathbf{p}_n$)
  $s \leftarrow s + 1$
  if $F(p_n) - \text{Gap}_{s,n} > \sqrt{T}$ *and* $|\Sigma|^n \leq B$ **then**
    $n \leftarrow 2n$
    $s \leftarrow 0$
    $\mathbf{p}_n \leftarrow \frac{1}{|\Sigma|}$
$n_{\max} \leftarrow n$.
$\mathbf{p}_n \leftarrow$ BinarySearch($[1, n_{\max}], F(p_n) - \text{Gap}_{\tau_{\text{opt}},n} \leq \sqrt{T}$)
return $\mathbf{p}_n$

even the optimal n-gram model for this n will cause an increase in the regret.

Let n^* be the smallest n such that $F(\mathbf{p}_n) - \text{Gap}_{\tau_{\text{opt}},n} \leq \sqrt{T}$ (or the smallest value that exceeds our budget). We can find this value in $\log(n^*)$ time using a two-stage process. In the first stage, we double n after every violation until we find an upper bound on n^*, which we denote by $n_{\max}$. In the second stage, we perform a binary search within $[1, n_{\max}]$ to determine n^*. Each stage takes $\log(n^*)$ iterations, and each iteration is the cost of running Prod-EG for that specific value of n. Thus, the overall complexity of the algorithm is $\mathcal{O}\left(\log(n^*)\text{Cost}(\text{Prod-EG})\right)$, where Cost(Prod-EG) is the cost of running Prod-EG once. Algorithm 5 presents the pseudocode for this algorithm, n-gramSelect.

1.7 Time-independent approximation of competitor automata

In the previous section, we introduced the technique of approximating the automaton accepting competitor sequences of length T, $\mathcal{C}_T$. Intersecting $\mathcal{C}$ with $\mathcal{S}_T$ for different T typically results in different approximation automata. Since each approximation requires solving a convex optimization problem, this can become computationally expensive.

In this section, we show how one can approximate the competitor set for different T using a single approximation. The key is to approximate the original automaton $\mathcal{C}$ directly. Specifically, assume first that $\mathcal{C}$ is a stochastic automaton (so that its outgoing transition weights at each state sum to 1). Let $\mathsf{p}_{\mathcal{C}}$ denote the distribution defined by $\mathcal{C}$, and let $\mathcal{P}$ be a family of distributions over Σ^* that we will use to approximate $\mathsf{p}_{\mathcal{C}}$. Given, $\mathsf{p}_{\mathcal{A}} \in \mathcal{P}$, define for every $x \in \Sigma^*$

$$\tilde{\mathsf{p}}_{\mathcal{A}}[x] = \begin{cases} \mathsf{p}_{\mathcal{A}}[x]\frac{\mathsf{p}_{\mathcal{C}}[\mathcal{S}_{|x|}]}{\mathsf{p}_{\mathcal{A}}[\mathcal{S}_{|x|}]} & \text{if } \mathsf{p}_{\mathcal{A}}[\mathcal{S}_{|x|}] > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Thus, $\tilde{\mathsf{p}}_{\mathcal{A}}[x]$ is a rescaling of $\mathsf{p}_{\mathcal{A}}$ based on the mass assigned by $\mathcal{C}$ to sequences of length equal to $|x|$. Note that $\tilde{\mathsf{p}}_{\mathcal{A}}$ may not necessarily be a distribution. Our algorithm consists of determining the best approximation to the competitor distribution $\mathsf{p}_{\mathcal{C}}$ within the family of rescaled distributions:

$$\min_{\mathsf{p}_{\mathcal{A}} \in \mathcal{P}} D_\infty(\mathsf{p}_{\mathcal{C}} || \tilde{\mathsf{p}}_{\mathcal{A}}). \tag{1.9}$$

Note that this is an implicit extension of the definition of ∞-Rényi divergence,

since $\tilde{\mathsf{p}}_{\mathcal{A}}$ is not a distribution.

The design of this optimization problem is motivated by the following result, which guarantees that if $\tilde{\mathsf{p}}_{\mathcal{A}}$ is a good approximation of $\mathsf{p}_{\mathcal{C}}$, then $\mathsf{p}_{\mathcal{A}\cap\mathcal{S}_T}$ will be a good approximation of $\mathsf{p}_{\mathcal{C}_T}$ for any T.

Theorem 1.7. *For any stochastic automata $\mathcal{S}$ and $\mathcal{A}$, and for any $T \geq 1$,*

$$D_\infty(\mathsf{p}_{\mathcal{C}_T}||\mathsf{p}_{\mathcal{A}\cap\mathcal{S}_T}) \leq D_\infty(\mathsf{p}_{\mathcal{C}}||\tilde{\mathsf{p}}_{\mathcal{A}}).$$

Proof. Let $x \in \mathcal{C}_T = \mathcal{C}\cap\mathcal{S}_T$ such that $\mathsf{p}_{\mathcal{C}_T}[x] > 0$. Since $\mathsf{p}_{\mathcal{C}_T}[x] = \frac{\mathsf{p}_{\mathcal{C}}[x]}{\mathsf{p}_{\mathcal{C}}(\mathcal{S}_T)}$, this implies that $\mathsf{p}_{\mathcal{C}}(\mathcal{S}_T) \geq \mathsf{p}_{\mathcal{C}}[x] > 0$. Thus, if $\mathsf{p}_{\mathcal{A}}(\mathcal{S}_T) > 0$, then

$$\log\left(\frac{\mathsf{p}_{\mathcal{C}_T}[x]}{\mathsf{p}_{\mathcal{A}\cap\mathcal{S}_T}[x]}\right) = \log\left(\frac{\mathsf{p}_{\mathcal{C}}[x]\mathsf{p}_{\mathcal{A}}(\mathcal{S}_T)}{\mathsf{p}_{\mathcal{C}}(\mathcal{S}_T)\mathsf{p}_{\mathcal{A}}[x]}\right) = \log\left(\frac{\mathsf{p}_{\mathcal{C}}[x]}{\tilde{\mathsf{p}}_{\mathcal{A}}[x]}\right).$$

On the other hand, if $\mathsf{p}_{\mathcal{A}}(\mathcal{S}_T) = 0$, then, by definition, $\tilde{\mathsf{p}}_{\mathcal{A}}[x] = 0$, therefore the following inequality holds

$$\log\left(\frac{\mathsf{p}_{\mathcal{C}_T}[x]}{\mathsf{p}_{\mathcal{A}\cap\mathcal{S}_T}[x]}\right) \leq \infty = \log\left(\frac{\mathsf{p}_{\mathcal{C}}[x]}{\tilde{\mathsf{p}}_{\mathcal{A}}[x]}\right).$$

The result now follows by taking the maximum over $x \in \mathcal{S}_T$ on the left-hand side and the maximum over $x \in \Sigma^*$ on the right-hand side. □

Note that, for n-gram approximations, $\mathsf{p}_{\mathcal{A}_n} \in \mathcal{P}_n$, the condition $\mathsf{p}_{\mathcal{A}_n}(\mathcal{S}_T) = 1$ always holds. Thus, the approximation optimization problem can be written as:

$$\min_{\mathsf{p}_{\mathcal{A}_n}\in\mathcal{P}_n} D_\infty(\mathsf{p}_{\mathcal{C}}||\tilde{\mathsf{p}}_{\mathcal{A}}) = \min_{\mathsf{p}_{\mathcal{A}_n}\in\mathcal{P}_n} \sup_{x\in\mathcal{C}} \log\left(\frac{\mathsf{p}_{\mathcal{C}}[x]}{\mathsf{p}_{\mathcal{A}_n}[x]\mathsf{p}_{\mathcal{C}}[\mathcal{S}_{|x|}]}\right).$$

As in Section 1.6, this problem is the minimization of the supremum of a family of

convex functions over the product of simplices. Thus, it is a convex optimization problem and can be solved using the PROD-EG algorithm.

We have thus far assumed that $\mathcal{C}$ is a stochastic automaton in this section. If the sum of the weights of all paths accepted by $\mathcal{C}$ is finite, we can apply weight-pushing to normalize the automaton to make it stochastic and then solve the approximation problem above.

However, this property may not always hold. For example, the original k-shifting automaton shown in Figure 1.2 accepts an infinite number of paths (sequences of arbitrary length with k shifts). Since each transition has unit weight, each path also has unit weight, and the sum of the weight of all paths is infinite.

However, we can still apply the approximation method in this section to the k-shifting automaton by rescaling the transitions weights of self-loops to be less than 1. Specifically, consider the automaton $\mathcal{C}_{k\text{-shift},\epsilon}$ whose states and transitions are exactly the same as those of the original automaton $\mathcal{C}_{k\text{-shift}}$, except that transitions from ta_j to $(t+1)a_k$ for $a_j \neq a_k$ now have weight $\frac{\epsilon}{N+1}$, and self-loops now have weight $1-\epsilon$. To make the automaton stochastic, we also assign weight $\frac{1}{N}$ to every initial state. Then, the weight of a path of length T accepted by $\mathcal{C}_{k\text{-shift},\epsilon}$ is $(1-\epsilon)^{T-k-1}\left(\frac{\epsilon}{N-1}\right)^k \frac{1}{N}$, and the weight of all paths is equal to the following:

$$\begin{aligned}
\sum_{\pi \in \mathcal{C}_{k\text{-shift},\epsilon}} w_{\mathcal{C}_{k\text{-shift},\epsilon}}[\pi] &= \sum_{T \geq k+1} N \binom{T-1}{k} (N-1)^k (1-\epsilon)^{T-k-1} \left(\frac{\epsilon}{N-1}\right)^k \frac{1}{N} \\
&= \sum_{T \geq k+1} \binom{T-1}{k} (1-\epsilon)^{T-k-1} \epsilon^k \\
&\leq \sum_{T \geq k+1} \left(\frac{(T-1)e}{k}\right)^k (1-\epsilon)^{T-k-1} \epsilon^k \\
&< +\infty.
\end{aligned}$$

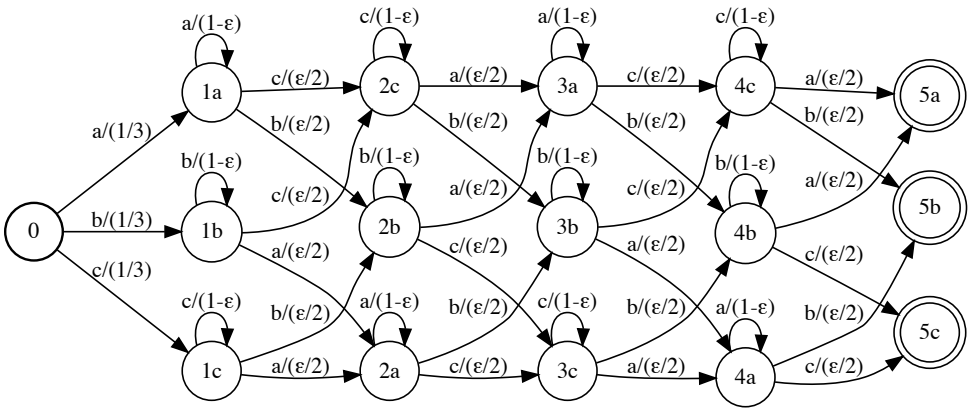


Figure 1.6: Illustration of $\mathcal{C}_{k\text{-shift},\epsilon}$, where $k = 4$ and $\Sigma = \{a, b, c\}$. This automaton accepts an infinite number of paths, and the weights have been rescaled from the original k-shifting automaton so that the path weights are now summable.

By normalizing the weights of this automaton, we can convert it into a stochastic automaton, where

$$\mathsf{p}_{\mathcal{C}_{k\text{-shift},\epsilon}}[\pi] \propto (1-\epsilon)^{|\pi|-k-1} \left(\frac{\epsilon}{N-1}\right)^k \frac{1}{N}.$$

Figure 1.6 shows the weighted automaton $\mathcal{C}_{k\text{-shift},\epsilon}$.

To compare with the results in Section 1.6, we will now analyze the approximation error of a maximum-likelihood-based bigram approximation.

Theorem 1.8 (Bigram approximation of $\mathcal{C}_{k\text{-shift},\epsilon}$)**.** *The maximum-likelihood based bigram model for $\mathcal{C}_{k\text{-shift},\epsilon}$ is defined by*

$$\mathsf{p}_{\mathcal{A}_2}[z_2|z_1] = \frac{\sum_{\tilde{T}\geq k+1}(1-\epsilon)^{\tilde{T}-k-1}\left(1_{z_1\neq z_2}\frac{k}{\tilde{T}-1}\frac{1}{N-1} + 1_{z_1=z_2}\left(1-\frac{k}{\tilde{T}-1}\right)\right)}{\sum_{\tilde{T}\geq k+1}(1-\epsilon)^{\tilde{T}-k-1}}.$$

Moreover, for every $T > k+1$, there exists $\epsilon \in (0,1)$ such that

$$D_\infty(\mathsf{p}_{\mathcal{C}_{k\text{-}shift,T}} || \mathsf{p}_{\mathcal{A}_2}) \leq -\log\left(1 - 2e^{-\frac{1}{12k}}\right).$$

Proof. The maximum-likelihood n-gram automaton is derived from the expected counts of the original automaton. Thus, for any $z_1, z_2 \in \Sigma$,

$$\begin{aligned}
\mathsf{p}_{\mathcal{A}_2}[z_2|z_1] &= \frac{\sum_{x \in \mathcal{C}_{k\text{-shift},\epsilon}} (1-\epsilon)^{|x|-k-1} \left(\frac{\epsilon}{N-1}\right)^k \frac{1}{N} \sum_{t=2}^{|x|} 1_{z_1^2 = x_{t-1}^t}}{\sum_{x \in \Sigma^*} (1-\epsilon)^{|x|-k-1} \left(\frac{\epsilon}{N-1}\right)^k \frac{1}{N} \sum_{t=2}^{|x|} 1_{z_1 = x_{t-1}}} \\
&= \frac{\sum_{x \in \mathcal{C}_{k\text{-shift},\epsilon}} (1-\epsilon)^{|x|-k-1} \sum_{t=2}^{|x|} 1_{z_1^2 = x_{t-1}^t}}{\sum_{x \in \Sigma^*} (1-\epsilon)^{|x|-k-1} \sum_{t=2}^{|x|} 1_{z_1 = x_{t-1}}} \\
&= \frac{\sum_{\tilde{T} \geq k+1} \sum_{x \in \mathcal{C}_{k\text{-shift},\tilde{T}}} (1-\epsilon)^{\tilde{T}-k-1} \sum_{t=2}^{\tilde{T}} 1_{z_1^2 = x_{t-1}^t}}{\sum_{\tilde{T} \geq k+1} \sum_{x \in \mathcal{C}_{k\text{-shift},\tilde{T}}} (1-\epsilon)^{\tilde{T}-k-1} \sum_{t=2}^{\tilde{T}} 1_{z_1 = x_{t-1}}}
\end{aligned}$$

Now, notice that for any $\tilde{T}$,

$$\begin{aligned}
&\sum_{x \in \mathcal{C}_{k\text{-shift},\tilde{T}}} \sum_{t=2}^{\tilde{T}} 1_{z_1^2 = x_{t-1}^t} \\
&\quad = \sum_{x \in \mathcal{C}_{k\text{-shift},\tilde{T}}} \sum_{t=2}^{\tilde{T}} 1_{z_1 = x_{t-1}} \left(1_{z_1 \neq z_2} \frac{k}{\tilde{T}-1} \frac{1}{N-1} + 1_{z_1 = z_2} \left(1 - \frac{k}{\tilde{T}-1}\right)\right).
\end{aligned}$$

This allows us to rewrite the probability above as:

$$\begin{aligned}
&\mathsf{p}_{\mathcal{A}_2}[z_2|z_1] \\
&= \frac{\sum_{\tilde{T}\geq k+1}(1-\epsilon)^{\tilde{T}-k-1}\left(1_{z_1\neq z_2}\frac{k}{\tilde{T}-1}\frac{1}{N-1}\right)\sum_{x\in\mathcal{C}_{k\text{-shift},\tilde{T}}}\sum_{t=2}^{\tilde{T}}1_{z_1=x_{t-1}}}{\sum_{\tilde{T}\geq k+1}(1-\epsilon)^{\tilde{T}-k-1}\sum_{x\in\mathcal{C}_{k\text{-shift},\tilde{T}}}\sum_{t=2}^{\tilde{T}}1_{z_1=x_{t-1}}} \\
&\quad + \frac{\sum_{\tilde{T}\geq k+1}(1-\epsilon)^{\tilde{T}-k-1}\left(1_{z_1=z_2}\left(1-\frac{k}{\tilde{T}-1}\right)\right)\sum_{x\in\mathcal{C}_{k\text{-shift},\tilde{T}}}\sum_{t=2}^{\tilde{T}}1_{z_1=x_{t-1}}}{\sum_{\tilde{T}\geq k+1}(1-\epsilon)^{\tilde{T}-k-1}\sum_{x\in\mathcal{C}_{k\text{-shift},\tilde{T}}}\sum_{t=2}^{\tilde{T}}1_{z_1=x_{t-1}}} \\
&= \frac{\sum_{\tilde{T}\geq k+1}(1-\epsilon)^{\tilde{T}-k-1}\left(1_{z_1\neq z_2}\frac{k}{\tilde{T}-1}\frac{1}{N-1}+1_{z_1=z_2}\left(1-\frac{k}{\tilde{T}-1}\right)\right)}{\sum_{\tilde{T}\geq k+1}(1-\epsilon)^{\tilde{T}-k-1}}.
\end{aligned}$$

Thus, $\mathsf{p}_{\mathcal{A}_2}[z_2|z_1]$ depends only on the condition $z_1 \neq z_2$.

Now, fix $T > k+1$. Since for every $x \in \mathcal{C}_{k\text{-shift},T} = \mathcal{C}_{k\text{-shift}} \cap \mathcal{S}_T$, x has k shifts and length T, $\mathsf{p}_{\mathcal{A}_2}$ is uniform over all sequences in $\mathcal{C}_{k\text{-shift},T}$. This allows us to bound the ∞-Renyi divergence between $\mathsf{p}_{\mathcal{C}_{k\text{-shift},T}}$ and $\mathsf{p}_{\mathcal{A}_2}$ by:

$$\begin{aligned}
&\sup_{x\in\mathcal{C}_{k\text{-shift},T}} \log\left(\frac{\mathsf{p}_{\mathcal{C}_{k\text{-shift},T}}[x]}{\mathsf{p}_{\mathcal{A}_2}[x]}\right) \\
&= \sup_{x\in\mathcal{C}_{k\text{-shift},T}} \log\left(\frac{\mathsf{p}_{\mathcal{C}_{k\text{-shift},T}}[x]}{\mathsf{p}_{\mathcal{A}_2}[\xi=x|\xi\in\mathcal{C}_T]\mathsf{p}_{\mathcal{A}_2}[\xi\in\mathcal{C}_T]+\mathsf{p}_{\mathcal{A}_2}[\xi=x|\xi\notin\mathcal{C}_{k\text{-shift},T}]\mathsf{p}_{\mathcal{A}}[\xi\notin\mathcal{C}_{k\text{-shift},T}]}\right) \\
&\leq \sup_{x\in\mathcal{C}_{k\text{-shift},T}} \log\left(\frac{\frac{1}{|\mathcal{C}_{k\text{-shift},T}|}}{\frac{1}{|\mathcal{C}_{k\text{-shift},T}|}\mathsf{p}_{\mathcal{A}_2}[\xi\in\mathcal{C}_{k\text{-shift},T}]}\right) \\
&= \sup_{x\in\mathcal{C}_{k\text{-shift},T}} \log\left(\frac{1}{\mathsf{p}_{\mathcal{A}_2}[\xi\in\mathcal{C}_{k\text{-shift},T}]}\right).
\end{aligned}$$

If we now let $(\xi_t)_{t=2}^T$ denote i.i.d. Bernoulli random variables with mean

$$\bar{p}(\epsilon) = \frac{\sum_{\tilde{T}\geq k+1}(1-\epsilon)^{\tilde{T}-k-1}\frac{k}{\tilde{T}-1}}{\sum_{\tilde{T}\geq k+1}(1-\epsilon)^{\tilde{T}-k-1}},$$

then

$$\begin{aligned}\mathsf{p}_{\mathcal{A}_2}[\xi \notin \mathcal{C}_{k\text{-shift},T}] &= \mathbb{P}\left[\left|\sum_{t=2}^{T} \xi_t - k\right| \geq 1\right] \\ &\leq \mathbb{P}\left[\left|\sum_{t=2}^{T} \xi_t - \bar{p}(\epsilon)(T-1)\right| \geq \frac{1}{2}\right] + \mathbb{P}\left[|\bar{p}(\epsilon)(T-1) - k| \geq \frac{1}{2}\right].\end{aligned}$$

Thus, if $|\bar{p}(\epsilon)(T-1) - k| < \frac{1}{2}$, then $\mathsf{p}_{\mathcal{A}_2}[\xi \notin \mathcal{C}_T]$ can be bounded using the same concentration argument as in Theorem 1.4.

$\bar{p}(\epsilon)$ can be interpreted as the weighted average of $\frac{k}{\tilde{T}-1}$ for $\tilde{T} \geq k+1$, where the weight of $\frac{k}{\tilde{T}-1}$ is $(1-\epsilon)^{\tilde{T}-k-1}$. We want this average to be close to $\frac{k}{T-1}$ for the specific choice of $T > k+1$, which we obtain by appropriately tuning $\epsilon \in (0,1)$.

Since $\lim_{\epsilon \to 0^+} \bar{p}(\epsilon) = 1$, $\lim_{\epsilon \to 1^-} \bar{p}(\epsilon) = 0$ and $\bar{p}(\epsilon)$ is continuous in ϵ on $(0,1)$, it follows by the intermediate value theorem that for any $T > k+1$, there exists an ϵ^* such that $\bar{p}(\epsilon^*) = \frac{k}{T-1}$. □

Note that in the proof of the above theorem, $\bar{p}(\epsilon)$ is monotonic in ϵ. Thus, one can find ϵ' such that $\left|\bar{p}(\epsilon') - \frac{k}{T-1}\right| \leq \frac{1}{2(T-1)}$ using binary search.

1.8 Failure transition algorithm

The computational complexity of the AWM algorithm presented in Section 1.3 is based on the size of the composed automaton $\mathcal{C} \cap \mathcal{S}_T$, which itself is related to the original size of $\mathcal{C}$. Similarly, if we were to apply AWM to an n-gram approximation, the computational complexity of the algorithm depends on the size of the approximating automaton. In this section, we introduce a technique to improve the computational cost of AWM by reducing the size of the automaton. To do this, we will use the notion of *failure transition (or φ-transition)*. Failure

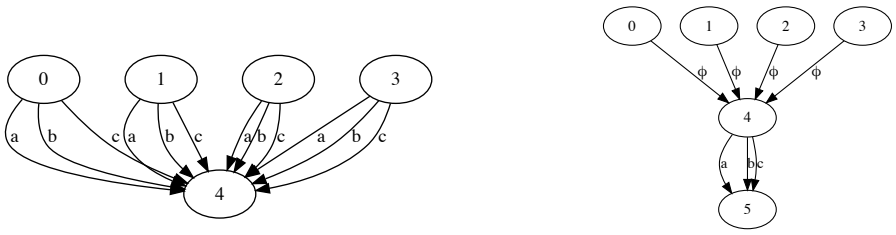
transitions play a key role in the design of many efficient string-matching and automata algorithms, including `egrep` under UNIX [Aho and Corasick, 1975, Knuth et al., 1977, Crochemore, 1986, Mohri, 1997].

Let φ be a symbol not in Σ. A φ-automaton is an automaton $\mathcal{A}$ with transition set $E_{\mathcal{A}} \subseteq Q_{\mathcal{A}} \times \Sigma \cup \{\varphi\} \cup \mathbb{R}_+ \cup Q_{\mathcal{A}}$. Transitions of the forms $e = (q, \varphi, w, q')$ are called failure transitions and characterized by the semantic of "other". If at state q there is no outgoing transition labeled with a given label $a \in \Sigma$ and there is a φ-transition e leaving q, then the failure transition is taken instead without consuming the label, and the next state is determined using the transitions leaving q'. We assume that there is no φ-cycle in any of our φ-automata, and that there is at most one failure transition leaving any state. This implies that the number of consecutive failure transitions taken is bounded.

More formally, the transition function δ is defined as follows in the presence of φ-transitions. Recall that for state $q \in Q_{\mathcal{A}}$ and label $a \in \Sigma$, $\delta[q, a]$ is the set of states reachable from q by reading label a, and $E[q, a]$ is the set of outgoing transitions from q with label a. For a φ-automaton, the set of states reachable from any state by reading label a is defined as follows:

$$\delta[q, a] = \begin{cases} \{\text{dest}[e] \colon e \in E[q, a]\} & \text{if } E[q, a] \neq \varnothing \\ \delta[\delta[q, \varphi], a] & \text{if } E[q, a] = \varnothing, E[q, \varphi] \neq \varnothing \\ \varnothing & \text{otherwise.} \end{cases}$$

A failure transition can often replicate the role of multiple standard transitions when there is "symmetry" within an automaton, that is when there are many transitions leading to the same state from different states that consume the same



(a) (b)

Figure 1.7: Example of the compression achieved by introducing a failure transition. (a) Standard automaton. (b) φ-automaton.

set of labels. Figure 1.7 illustrates such cases.

Given any non-initial state q in an automaton, we would like to determine a set of parent states within $E^R[q]$ from which it is beneficial to introduce a φ-transition. We design a greedy solution in Algorithm 6, φ-SourceSubset. For every $q \in Q \setminus I$, having already computed Q_{k-1}, φ-SourceSubset greedily determines a candidate set Q_k of k parent states based on which available parent state has the most label-weight overlap with the subset already chosen. It then selects the best candidate set from $\{Q_k\}_{k=1}^{|E^R[q]|}$. φ-SourceSubset is able to recover the φ-transition structure in Figure 1.7, and while it does not always produce the optimal subset of parent states, its computational cost is only $\mathcal{O}\big(|\delta^R[q]|^2|E^R[q]|\big)$ for input state q.

We now present an algorithm, φ-Convert, that converts a weighted automaton to a weighted φ-automaton without ever increasing the size of the automaton. Its pseudocode is provided in Algorithm 7, and it uses φ-SourceSubset as a subroutine to select a candidate set of parent states for each state q. The algorithm then determines whether it should introduce a failure transition between these parent states and q based on whether the φ-transition will reduce the total number

Algorithm 6: φ-SourceSubset.

Algorithm: φ-SourceSubset$(\mathcal{C}, q)$

$(S_0, Q_0) \leftarrow (\varnothing, \varnothing)$
$k^* \leftarrow 1$
for $k \leftarrow 1$ *to* $|\delta^R[q]|$ **do**
  $q_k \leftarrow$
  $\operatorname{argmax}_{q' \in \delta^R[q] \setminus Q_{k-1}} \left|(a, w) \in \Sigma \times \mathbb{R}_+ \colon \forall \tilde{q} \in \delta^R[q] \cup \{q'\}, (\tilde{q}, a, w, q) \in E_{\mathcal{C}}\right|$
  $S_k \leftarrow \left|(a, w) \in \Sigma \times \mathbb{R}_+ \colon \forall \tilde{q} \in \delta^R[q] \cup \{q_k\}, (\tilde{q}, a, w, q) \in E_{\mathcal{C}}\right|$
  $Q_k \leftarrow Q_{k-1} \cup \{q_k\}$
  $k^* \leftarrow \operatorname{argmax}_{j \in \{k, k^*\}} \{|S_j||Q_j| - (|S_j| + |Q_j|)\}$
return (S_{k^*}, Q_{k^*})

Algorithm 7: φ-Convert.

Algorithm: φ-Convert$(\mathcal{C})$

for each $q \in Q_{\mathcal{C}} \setminus I_{\mathcal{C}}$ **do**
  $S^*, Q^* \leftarrow \varphi$-SourceSubset$(\mathcal{C}, q)$
  if $|S^*| + |Q^*| < |S^*||Q^*|$ **then**
    $\tilde{q} \leftarrow$ NewState$(\mathcal{C})$
    $E_{\mathcal{C}} \leftarrow E_{\mathcal{C}} \cup \{(q, \varphi, 1, \tilde{q})\}$
    for each $q' \in Q^*$ **do**
      for each $e' \in E_{\mathcal{C}}[q']$ **do**
        if $(\sigma[e'], w[e']) \in S^*$ **then**
          $E_{\mathcal{C}} \leftarrow E_{\mathcal{C}} \cup \{(\tilde{q}, \sigma[e'], w[e'], q)\}$
          Delete$[E_{\mathcal{C}}, e']$

of transitions in the automaton. Theorem 1.9 provides an accompanying guarantee on efficiency.

Theorem 1.9 (Correctness and performance of φ-Convert)**.** φ-Convert *returns a φ-automaton equivalent to the input automaton. The number of transitions in the φ-automaton output by the algorithm is never larger than the number of transitions in the input automaton. The running-time complexity of* φ-Convert *is* $\mathcal{O}\Big(|Q_{\mathcal{C}}| \max_{q \in Q_{\mathcal{C}}} |\delta^R[q]|^2 |E^R[q]|\Big)$.

Proof. Let q be any state in $\mathcal{C} \setminus I_{\mathcal{C}}$, $q' \in \delta^R[q]$, and $e = (q', a, w, q) \in E^R[q]$ be such that we introduce a failure transition from q' to q that replaces e. Then, any path

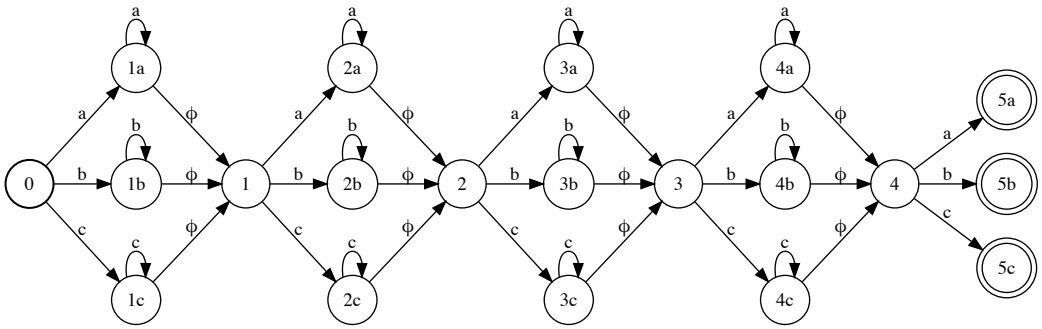


Figure 1.8: The k-shifting automaton converted into a φ-automaton. The new representation has a dramatic reduction in the number of transitions.

that used to contain transition e will now contain the consecutive pair of transitions $(q', \varphi, 1, \tilde{q})$ and $(\tilde{q}, a, w, q)$, which is taken where a was previously taken (since the automaton is deterministic). The contributed weight of this pair is w and the substring consumed is a, which is the same as it was for e. Since q was arbitrary, the new automaton assigns the same weight to every string as the old automaton.

Moreover, notice that whenever we consider introducing a failure transition, we remove $|S^*|$ transitions from $|Q^*|$ parent states while introducing $|Q^*|$ φ-transitions from each of the parent states to a new state $\tilde{q}$, and $|S^*|$ transitions from $\tilde{q}$ to q. Thus, the change in number of transitions is $|S^*| + |Q^*| - |S^*||Q^*|$. This is precisely the condition that we check before introducing a failure transition.

The running-time complexity of φ-Convert is dominated by the subroutine φ-SourceSubset. Since we must run it for every state in $Q_{\mathcal{C}} \setminus I_{\mathcal{C}}$, the total computational cost is $\mathcal{O}\big(|Q_{\mathcal{C}}| \max_{q \in Q_{\mathcal{C}}} |\delta^R[q]|^2 |E^R[q]|\big)$. □

Figure 1.8 shows the result of applying φ-Convert to the k-shifting automaton.

Recall that the two main automata operations required for AWM are composition and shortest-distance. While these two algorithms are standard for weighted automata, it is not as clear how one can perform them over weighted φ-automata. We now extend both to φ-automata.

First, we consider the composition of a weighted φ-automaton with $\mathcal{S}_T$. Algorithm 8 presents the pseudocode for our algorithm, φ-Composition.[3] As with standard composition the output automaton is based on matching transitions of $\mathcal{C}$ and $\mathcal{S}_T$. However, when the algorithm considers a φ-transition in $\mathcal{C}$, it checks if the label of the current transition considered in $\mathcal{S}_T$ matches that of a transition leaving the source state of the φ-transition. In that case, the algorithm skips this pair of transitions, as the φ-transition is not taken in the presence of transitions with a desired label. If there is no such label, then the algorithm creates a new transition for the composed automaton, where the destination state in the $\mathcal{S}_T$ component remains unchanged (since the automaton has not successfully "read" the label yet). Theorem 1.10 presents consistency guarantees that φ-composition behaves in an analogous way to standard composition.

Theorem 1.10 (φ-Composition consistency guarantee)**.** φ-Composition *outputs a φ-automaton equivalent to $\mathcal{C} \cap \mathcal{S}_T$ (or $\mathcal{C} \circ \mathcal{S}_T$). The maximum number of transitions created by the algorithm is $\mathcal{O}\Big(\sum_{t=1}^{T} |Q_{\mathcal{C},t}| \max_{q \in Q_{\mathcal{C},t}} |E[q]|\Big)$, where $Q_{\mathcal{C},t} = \{q \in Q_{\mathcal{C}} \times Q_{\mathcal{S}_T} \colon q = (\cdot, t)\}$.*

Proof. Given an automaton $\mathcal{A}$, state $q \in Q_{\mathcal{A}}$, and string $x \in \Sigma^*$, let $\delta_{\mathcal{A}}[q, w]$ be the set of states reachable from q by reading string x.

Let $q \in \mathcal{C}_T$. By construction, $q = (q_1, q_2)$, where $q_1 \in \mathcal{C}$ and $q_2 \in \mathcal{S}_T$. We first show that for any string $x \in \Sigma^*$, $\delta_{\mathcal{C}_T}[q, x] = (\delta_{\mathcal{C}}[q_1, x], \delta_{\mathcal{S}_T}[q_2, x])$.

[3] In the pseudocode, $\uplus$ denotes the multiset union.

Algorithm 8: φ-COMPOSITION.

Algorithm: φ-COMPOSITION($\mathcal{C}$, $\mathcal{S}_T$)

$Q_{\mathcal{C}_T} \leftarrow I_{\mathcal{C}} \times I_{\mathcal{S}_T}$.
$\mathcal{Q} \leftarrow I_{\mathcal{C}} \times I_{\mathcal{S}_T}$. **while** $\mathcal{Q} \neq \varnothing$ **do**
  $q = (q_1, q_2) \leftarrow \text{HEAD}(\mathcal{Q})$
  DEQUEUE($\mathcal{Q}$)
  if $q \in I_{\mathcal{C}} \times I_{\mathcal{S}_T}$ **then**
    $I_{\mathcal{C}_T} \leftarrow I_{\mathcal{C}_T} \cup \{q\}$
    $\lambda_{\mathcal{C}_T} \leftarrow \lambda_{\mathcal{C}}(q_1)\lambda_{\mathcal{S}_T}(q_2)$
  if $q \in F_{\mathcal{C}} \times F_{\mathcal{S}_T}$ **then**
    $F_{\mathcal{C}_T} \leftarrow F_{\mathcal{C}_T} \cup \{q\}$
    $\rho_{\mathcal{C}_T} \leftarrow \rho_{\mathcal{C}}(q_1)\rho_{\mathcal{S}_T}(q_2)$
  for each $(e_1, e_2) \in E_{\mathcal{C}} \times E_{\mathcal{S}_T}$ **do**
    if $\sigma(e_1) = \sigma(e_2)$ **then**
      if $\tilde{q} = (\text{dest}[e_1], \text{dest}[e_2]) \notin Q_{\mathcal{C}_T}$ **then**
        $Q_{\mathcal{C}_T} \leftarrow Q_{\mathcal{C}_T} \cup \{\tilde{q}\}$
        ENQUEUE($\mathcal{Q}, \tilde{q}$)
      $E_{\mathcal{C}_T} \leftarrow E_{\mathcal{C}_T} \uplus \{(q, \sigma(e_1), w[e_1]w[e_2], \tilde{q})\}$
    else if $\sigma[e_1] = \varphi$ **then**
      if $\nexists e \in E_{\mathcal{C}}[q_1]$ *such that* $\sigma[e] = \sigma[e_2]$ **then**
        if $\tilde{q} = (\text{dest}[e_1], q_2) \notin Q_{\mathcal{C}_T}$ **then**
          $Q_{\mathcal{C}_T} \leftarrow Q_{\mathcal{C}_T} \cup \{\tilde{q}\}$
          ENQUEUE($\mathcal{Q}, \tilde{q}$)
        $E_{\mathcal{C}_T} \leftarrow E_{\mathcal{C}_T} \uplus \{(q, \varphi, w[e_1], \tilde{q})\}$
$\mathcal{C}_T \leftarrow (\Sigma, Q_{\mathcal{C}_T}, I_{\mathcal{C}_T}, F_{\mathcal{C}_T}, E_{\mathcal{C}_T}, \lambda_{\mathcal{C}_T}, \rho_{\mathcal{C}_T})$
return $\mathcal{C}_T$

By induction, suppose that $|x| = 1$. Let $(q_1', q_2') \in (\delta_{\mathcal{C}}[q_1, x], \delta_{\mathcal{S}_T}[q_2, x])$. Then $\exists e_2 \in E_{\mathcal{S}_T}$ such that $e_2 = (q_2, x, w_2, q_2')$. Since $\mathcal{C}$ admits failure transitions, it is possible that $\exists e_1 \in E_{\mathcal{C}}$ such that $e_1 = (q_1, x, w_1, q_1')$ or not. If such an e_1 does exist, then φ-COMPOSITION constructs the edge $e = ((q_1, q_2), x, w_1w_2, (q_1', q_2'))$ so that $(q_1', q_2') \in E_{\mathcal{C}_T}$. If such an e_1 does not exist, then there must exist a sequence of failure transitions $\{\tilde{e}_j\}_{j=1}^{n}$ leading to a state with a transition $\tilde{e}_{n+1}$ labeled with x with destination q_1'. Moreover, there cannot be any transition leaving $\{q_1\} \cup \{\text{dest}[\tilde{e}_j]\}_{j=1}^{n-1}$ labeled with x. In this case, φ-COMPOSITION would have

constructed the transitions:

$$\{((q_1,q_2),\varphi,w[\tilde{e}_1],(\mathrm{dest}[\tilde{e}_1],q_2))\}\cup\{((\mathrm{dest}[\tilde{e}_j],q_2),\varphi,w[\tilde{e}_1],(\mathrm{dest}[\tilde{e}_{j+1}],q_2))\}_{j=1}^{n-1}$$
$$\cup\{((\mathrm{dest}[\tilde{e}_n],q_2),x,w[\tilde{e}_{n+1}]w_2,(q_1',q_2'))\},$$

and there also would be no transitions labeled with x leaving $\{(q_1,q_2)\}\cup\{\mathrm{dest}[\tilde{e}_j],q_2\}_{j=1}^{n}$. Thus, we would have $(q_1',q_2')\in\delta_{\mathcal{C}_T}[q,x]$. Since (q_1,q_2), (q_1',q_2') and x are arbitrary, this implies that $\delta_{\mathcal{C}_T}[q,x]\supseteq(\delta_{\mathcal{C}}[q_1,x],\delta_{\mathcal{S}_T}[q_2,x])$.

To show the other inclusion, assume that $(q_1',q_2')\in\delta_{\mathcal{C}_T}[q,x]$. Then, either there exists a transition of the form $((q_1,q_2),x,w,(q_1',q_2'))$ or not. If there does exist such a transition, then it must have been constructed from the transitions (q_1,x,w_1,q_1') and (q_2,x,w_2,q_2') in $E_{\mathcal{C}}$ and $E_{\mathcal{S}_T}$. Otherwise, the algorithm would have constructed failure transitions. Thus, if such a transition exists, then $q_1'\in E_{\mathcal{C}}[q_1,x]$ and $q_2'\in E_{\mathcal{S}_T}[q_2,x]$. If such a transition does not exist, then (q_1',q_2') is reached through a series of failure transitions. Thus, there must be a sequence of transitions:

$$\{((q_1,q_2),\varphi,\tilde{w}_1,\tilde{q}_1)\}\cup\{(\tilde{q}_j,\varphi,\tilde{w}_j,\tilde{q}_{j+1})\}_{j=2}^{n}\cup\{(\tilde{q}_{n+1},x,\tilde{w}_{n+1},(q_1',q_2'))\},$$

in which none of the states $\{(q_1,q_2)\}\cup\{\tilde{q}_j\}_{j=1}^{n}$ have outgoing transitions labeled with x. By construction, each of the states $\tilde{q}_j=(\tilde{q}_{j,1},\tilde{q}_{j,2})\in Q_{\mathcal{C}}\times Q_{\mathcal{S}_T}$, and any failure transition in the output: $\forall j\in[2,n]$, $(\tilde{q}_j,\varphi,\tilde{w}_j,\tilde{q}_{j+1})$ must have been constructed from the transition $(\tilde{q}_{j,1},\varphi,\tilde{w}_j,\tilde{q}_{j+1,1})\in E_{\mathcal{C}}$, and satisfy $\tilde{q}_{j,2}=\tilde{q}_{j+1,2}$. Similarly, there must exist $(q_1,\varphi,\tilde{w}_1,\tilde{q}_{1,1})\in E_{\mathcal{C}}$ and $\tilde{q}_{1,2}=q_2$. Finally, the existence of $(\tilde{q}_{n+1},x,\tilde{w}_{n+1},(q_1',q_2'))$ implies the existence of both $(\tilde{q}_{n+1,1},x,\tilde{w}_{n+1,1},q_1')\in E_{\mathcal{C}}$ and $(\tilde{q}_{n+1,2},x\tilde{w}_{n+1,2},q_2')\in E_{\mathcal{S}_T}$, where $\tilde{w}_{n+1,1}\tilde{w}_{n+1,2}=\tilde{w}_{n+1}$. In other words, $E_{\mathcal{S}_T}$ contains a transition $(q_2,x,\tilde{w}_{n+1,2},q_2')$ and $E_{\mathcal{C}}$ contains a sequence of failure transitions whose

source states do not have outgoing edges labeled with x. But this means that the destination of this sequence of transitions (q_1', q_2') is reachable from (q_1, q_2) by reading x, and so $\delta_{\mathcal{C}_T}[q, x] \subseteq (\delta_{\mathcal{C}}[q_1, x], \delta_{\mathcal{S}_T}[q_2, x])$. Thus, $\delta_{\mathcal{C}_T}[q, x] = (\delta_{\mathcal{C}}[q_1, x], \delta_{\mathcal{S}_T}[q_2, x])$ when $|x| = 1$.

By induction, assume that the statement is true for $|x| = n - 1$ and that now $|x| = n$. Then $x = yz$ where $|y| = n - 1$ and $|z| = 1$. Thus, we have

$$\begin{aligned}
\delta_{\mathcal{C}_T}[q, x] &= \delta_{\mathcal{C}_T}[q, yz] \\
&= \delta_{\mathcal{C}_T}[\delta_{\mathcal{C}_T}[q, y], z] \\
&= (\delta_{\mathcal{C}}[\delta_{\mathcal{C}}[q_1, y], z], \delta_{\mathcal{S}_T}[\delta_{\mathcal{S}_T}[q_2, y], z]) \\
&= (\delta_{\mathcal{C}}[q_1, yz], \delta_{\mathcal{S}_T}[q_2, yz]) \\
&= (\delta_{\mathcal{C}}[q_1, x], \delta_{\mathcal{S}_T}[q_2, x]).
\end{aligned}$$

Thus, $\delta_{\mathcal{C}_T}[q, x] = (\delta_{\mathcal{C}}[q_1, x], \delta_{\mathcal{S}_T}[q_2, x])$ for all $x \in \Sigma^*$.

Notice that in the course of the proof above, we have constructed a one-to-one correspondence between the paths in $E_{\mathcal{C}_T}$ and the paths in $E_{\mathcal{C}}$ and $E_{\mathcal{S}_T}$, where the weights of the paths in the former are exactly the product of the weights of the paths in the latter. Since the weight of any string is simply the sum of the weights of all accepting paths, the weight of any string in the output is equal to the sum of the products of the weights of matching paths in $\mathcal{C}$ and $\mathcal{S}_T$ in the input.

The bound on the number of transitions in the output of composition follows from the same reasoning as for standard composition. Since the algorithm only constructs states and transitions that are reachable by paths in $\mathcal{C} \cap \mathcal{S}_T$, we may again consider the incremental cost of composing states corresponding to $q = (\cdot, t + 1)$ given that we have composed all states of the form $q = (\cdot, t)$. The existence of

failure states and transitions is accounted for in the definition of $Q_{\mathcal{C},t}$.

□

The second automaton operation that we need to extend to φ-automata is the shortest-distance computation. Specifically, in AWM, we applied Algorithm 2, IncrSD, as a subroutine, which was a shortest-distance computation for weighted automata defined over the probability semiring. In general, IncrSD cannot be directly applied to φ-automata because it might sum over potentially 'obsolete φ-transitions'. For example, if at a given state q, there is a transition labeled with a to q' and a φ-transition whose destination state has a single outgoing transition also labeled with a to q', the second path should not be considered.

To account for these types of situations and extend IncrSD in a computationally efficient way, we use the fact that the probability semiring $(\mathbb{R}_+, +, \times, 0, 1)$ admits a natural extension to a ring structure under the standard additive inverse -1. Specifically, upon encountering a transition e labeled with a leaving state q, we will check for φ-transitions with destination states that admit further transitions e' labeled with a. Any such transition should not contribute any weight to the distance to dest$[e']$, since under the semantic of the φ-transition, this path should never be taken. To correctly account for these paths, we will preemptively subtract the weight of e' from its destination state. When the algorithm processes the φ-transition directly, it will add this weight back so that the total contribution of this path is zero. The overhead of this extra computation is $\mathcal{O}(N_\varphi(Q_{\mathcal{C}_T}) \max_{q \in Q_{\mathcal{C}_T}} |E[q]|)$, where $N_\varphi(Q_{\mathcal{C}_T})$ is the maximum number of consecutive φ-transitions leaving states in $Q_{\mathcal{C}_T}$. The precise pseudocode is presented in Algorithm 9, φ-IncrSD. Theorem 1.11 provides a theoretical guarantee for the algorithm.

Algorithm 9: φ-IncrementalShortestDistance (φ-IncrSD).

Algorithm: φ-IncrSD($\mathcal{C} \cap \mathcal{S}_T$, t, $\boldsymbol{\alpha}$)
$\mathcal{C}_T \leftarrow \mathcal{C} \cap \mathcal{S}_T$
for each $q \in Q_{\mathcal{C}_T}$ **with** $q = (\cdot, t)$ **do**
  $\alpha[q] \leftarrow r[q] \leftarrow 0$
if $t = 1$ **then**
  for each $q \in I_{\mathcal{C}_T}$ **do**
    $\alpha[q] \leftarrow r[q] \leftarrow 1$
  $\mathcal{Q} \leftarrow I_{\mathcal{C}_T}$.
else
  $\mathcal{Q} \leftarrow \{q \in Q_{\mathcal{C}_T} : q = (\cdot, t-1)\}$
while $\mathcal{Q} \neq \varnothing$ **do**
  $q \leftarrow \text{Head}(\mathcal{Q})$
  Dequeue($\mathcal{Q}$)
  $\tilde{r} \leftarrow r[q]$
  $r[q] \leftarrow 0$
  for each $e \in E_{\mathcal{C}_T}[q]$ **do**
    if $\alpha[\text{dest}[e]] \neq \alpha[\text{dest}[e]] + (\tilde{r}w[e])$ **then**
      $\alpha[\text{dest}[e]] \leftarrow \alpha[\text{dest}[e]] + (\tilde{r}w[e])$
      $r[\text{dest}[e]] \leftarrow r[\text{dest}[e]] + (\tilde{r}w[e])$
      if $\text{dest}[e] \notin \mathcal{Q}$ **then**
        Enqueue($\mathcal{Q}$, dest[e])
    if $\sigma[e] \neq \varphi$ **then**
      $\tilde{q} \leftarrow q$
      $w_\varphi \leftarrow 1$
      while $\exists e_\varphi \in E[\tilde{q}]$ **with** $\sigma[e_\varphi] = \varphi$ **do**
        $w_\varphi \leftarrow w_\varphi w[e_\varphi]$
        if $\exists e' \in E[\text{dest}[e_\varphi]]$ **with** $\sigma[e'] = \sigma[e]$ **then**
          $\alpha[\text{dest}[e']] \leftarrow \alpha[\text{dest}[e']] - (\tilde{r}w_\varphi w[e'])$
          Break
        else
          $\tilde{q} \leftarrow \text{dest}[e_\varphi]$
return $\boldsymbol{\alpha}$

Theorem 1.11 (φ-IncrSD guarantee)**.** *If* φ-IncrSD *is run with the topological order queue discipline and the shortest distances from source* $I_{\mathcal{C}_T}$ *to the set of states* $Q_{\mathcal{C},t-1} = \{q \in Q_{\mathcal{C}_T} : q = (\cdot, t-1)\}$ *are stored in* $\boldsymbol{\alpha}$*, then* φ-IncrSD *computes the shortest distances from source* $I_{\mathcal{C}_T}$ *to the set of states* $Q_{\mathcal{C},t}$*. The computational*

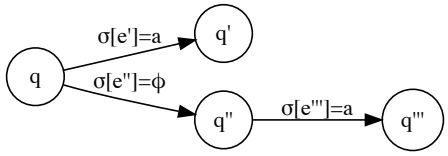


Figure 1.9: Illustration of φ-IncrSD. While processing state q, the algorithm pre-emptively reverses the future relaxation at q''': $\alpha[q'''] \leftarrow \alpha[q'''] - \tilde{r}w[e_\varphi]w[e''']$.

complexity of φ-IncrSD *is* $\mathcal{O}\Big(N_\varphi(Q_{\mathbb{e},t-1})|E[Q_{\mathbb{e},t-1}]|\Big)$.

Proof. Assume that $\boldsymbol{\alpha}$ stores the shortest distances from $I_{\mathbb{e}_T}$ to $Q_{\mathbb{e},t-1}$, and let $q \in Q_{\mathbb{e},t}$. Since we assume that $\boldsymbol{\alpha}$ is correct up to $Q_{\mathbb{e},t-1}$ and every path from $I_{\mathbb{e}_T}$ to $Q_{\mathbb{e},t}$ must pass through some state in $Q_{\mathbb{e},t-1}$, it follows that

$$\begin{aligned}\alpha[q] &= \sum_{\substack{\pi \in P[Q_{\mathbb{e},t-1},q] \\ \pi \text{ valid}}} \alpha[\mathrm{src}[\pi]]w[\pi] \\ &= \sum_{\pi \in P[Q_{\mathbb{e},t-1}]} \alpha[\mathrm{src}[\pi]]w[\pi] - \sum_{\substack{\pi \in P[Q_{\mathbb{e},t-1},q] \\ \pi \text{ invalid}}} \alpha[\mathrm{src}[\pi]]w[\pi],\end{aligned}$$

where a valid π refers to a path that respects the semantic of the φ-transition.

The first sum in the last expression is the value computed by the standard shortest-distance algorithm, IncrSD, ignoring the semantic of the φ-transition. It is also the value computed by the algorithm in the first "if" clause within the loop over $E_{\mathbb{e}_T}[q]$. The second sum is the value computed and subtracted out from $\alpha[q]$ by the second "if" clause within the same loop, when we encounter a transition e such that $\sigma[e] \neq \varphi$. Thus, $\alpha[q]$ is computed correctly, and since $q \in Q_{\mathbb{e},t}$ is arbitrary, φ-IncrSD returns $\boldsymbol{\alpha}$ with the correct path weights for all states in $Q_{\mathbb{e},t}$.

Notice that the algorithm loops over every state in $Q_{\mathfrak{C},t-1}$, including states that are the destination of φ-transitions. For every state q, the algorithm loops over every transition $e \in E[q]$ and possibly looks into $N_\varphi(q)$ consecutive φ-transitions. If q' is the destination of a φ-transition, then this will cost $\mathcal{O}(E[q'])$. Thus, the total computational cost is $\mathcal{O}\left(N_\varphi(Q_{\mathfrak{C},t-1})|E[Q_{\mathfrak{C},t-1}]|\right)$. □

Notice that φ-IncrSD can be straightforwardly extended to a non-incremental form that computes the shortest path to every state given a source state. We call this algorithm φ-ShortestDistance, which we will use to compute the analogue of $\boldsymbol{\beta}$ in AutomataWeightedMajority.

With these two new operations, φ-Composition and φ-IncrSD, we are now ready to present φ-AutomataWeightedMajority (φ-AWM), an extension to AWM that applies to φ-automata. Given an input automaton (not necessarily with φ-transitions), the algorithm first calls φ-Convert to determine whether it is beneficial to introduce φ-transitions. The algorithm then composes the output with $\mathcal{S}_T$ to compute the set of sequences of length T that are accepted by $\mathcal{C}$. The algorithm then uses the new shortest-distance algorithm to perform the same weighted majority update as before. Algorithm 10 presents the pseudocode for φ-AWM.

Since the update in φ-AWM is identical to the update in AWM, we immediately obtain the following guarantee for φ-AWM.

Theorem 1.12 (φ-AutomataWeightedMajority Guarantee)**.** *If $\mathcal{C} \cap \mathcal{S}_T$ is normalized in the input to* PBWM*, then at each round t, φ-*AWM *performs the same weighted majority update as in* PBWM *for unweighted regret. Moreover, the computational complexity of φ-*AWM *at each round t is $\mathcal{O}\left(|N_\varphi(Q_{\mathfrak{e}_T,t-1})E[Q_{\mathfrak{e}_T,t}]|\right)$,*

Algorithm 10: φ-AutomataWeightedMajority (φ-AWM).

Algorithm: φ-AWM($\mathcal{C}$)

$\mathcal{C} \leftarrow \varphi$-Convert($\mathcal{C}$)
$\mathcal{C}_T \leftarrow \varphi$-Composition($\mathcal{C}, \mathcal{S}_T$)
$\mathcal{C}_T \leftarrow$ Connect($\mathcal{C}_T$)
$\boldsymbol{\beta} \leftarrow \varphi$-ShortestDistance($(\mathcal{C}_T)^R, F$)
$w_{\mathcal{C}_T}[I_{\mathcal{C}_T}] \leftarrow w_{\mathcal{C}_T}[I_{\mathcal{C}_T}]/\beta[I_{\mathcal{C}_T}]$
$\boldsymbol{\alpha} \leftarrow 0$
$\mathbf{Flow}_0 \leftarrow 0$
for each $e \in E_{\mathcal{C}_T}$ **with** $\mathrm{dest}[e] = (\cdot, 1)$ **do**
  $\mathrm{flow}[e] \leftarrow \beta[\mathrm{dest}[e]]$
  $\mathrm{Flow}_0[\sigma[e]] \leftarrow \mathrm{Flow}_0[\sigma[e]] + \mathrm{flow}[e]$
  $Z_0 \leftarrow Z_0 + \mathrm{flow}[e]$
$\mathsf{p}_1 \leftarrow \mathrm{Flow}_0/Z_0$
for $t \leftarrow 1$ **to** T **do**
  $i_t \leftarrow$Sample(p_t)
  Play(i_t)
  Receive($\mathbf{l}_t$)
  $\mathbf{Flow}_t \leftarrow 0$
  for each $e \in E_{\mathcal{C}_T}$ **with** $\mathrm{dest}[e] = (\cdot, t)$ **do**
    $w[e] \leftarrow w[e] \cdot e^{-\eta l_t[\sigma[e]]}$
    $\mathrm{flow}[e] \leftarrow \alpha[\mathrm{src}[e]] w[e] \beta[\mathrm{dest}[e]]$
    $\mathrm{Flow}_t[\sigma[e]] \leftarrow \mathrm{Flow}_t[\sigma[e]] + \mathrm{flow}[e]$
    $Z_t \leftarrow Z_t + \mathrm{flow}[e]$.
  $\mathsf{p}_{t+1} \leftarrow \mathbf{Flow}_t/Z_t$
  $\boldsymbol{\alpha} \leftarrow \varphi$-IncrSD($\mathcal{C}_T, t, \boldsymbol{\alpha}$)

so that the total computational cost is $\mathcal{O}\left(\sum_{t=1}^{T} N_\varphi(Q_{\mathcal{C}_T,t-1}) |E[Q_{\mathcal{C}_T,t}]|\right)$.

For the k-shifting automaton, the per-iteration computational complexity of φ-AWM is now $\mathcal{O}(Nk)$, since there is at most one consecutive φ-transition in the output of φ-Convert, and we now aggregate transitions at each time using failure transitions. This is a factor of N better than that of AWM, and only a factor of k worse than the specific algorithm of Herbster and Warmuth [1998]. Using a bigram approximation of the k-shifting automaton composed with $\mathcal{S}_T$ and then introducing converting it into a φ-automaton, we obtain an algorithm that runs in $\mathcal{O}(N)$.

1.9 Composition of multiple φ-automata

In many cases, it may be more convenient and, in some instances, more efficient to define $\mathcal{C}$ as the composition (intersection) of multiple automata, each represented with failure transitions. While the composition algorithms discussed above are correct for composing an automaton with failure transitions with an automaton without failure transitions, they are not guaranteed to be accurate when composing two φ-automata. Specifically, in the process of matching transitions, the composition algorithm may produce multiple φ-paths between two states. See Figure 1.10 for an example.

Having redundant φ-paths is problematic because it can lead to incorrect weight computations. Specifically, we may associate extra paths to a given string due to multiple φ-paths. As a consequence, we may also assign extra weight to this string.

To avoid this situation, we introduce the concept of a *φ-filter*, which is a mechanism that can filter out all but one φ-path between any two states. As we shall see, this φ-filter can be implemented as composition with a particular *finite-state transducer (FST)*.

Recall that a finite-state transducer is a finite automaton in which each transition is augmented with an output label. We denote the input label of a transition e as $i[e]$, the output label as $o[e]$, and the label pair as $(i[e] : o[e])$. The output labels are concatenated along a path to form an output sequence, so that any accepting path has an associated output string. A finite automaton can be interpreted as a finite-state transducer whose input and output labels coincide for each transition. As with finite automata, finite-state transducers can also be augmented with weights, and as with weighted finite automata, weighted finite-state transducers can be composed efficiently and on-demand [Mohri, 2009].

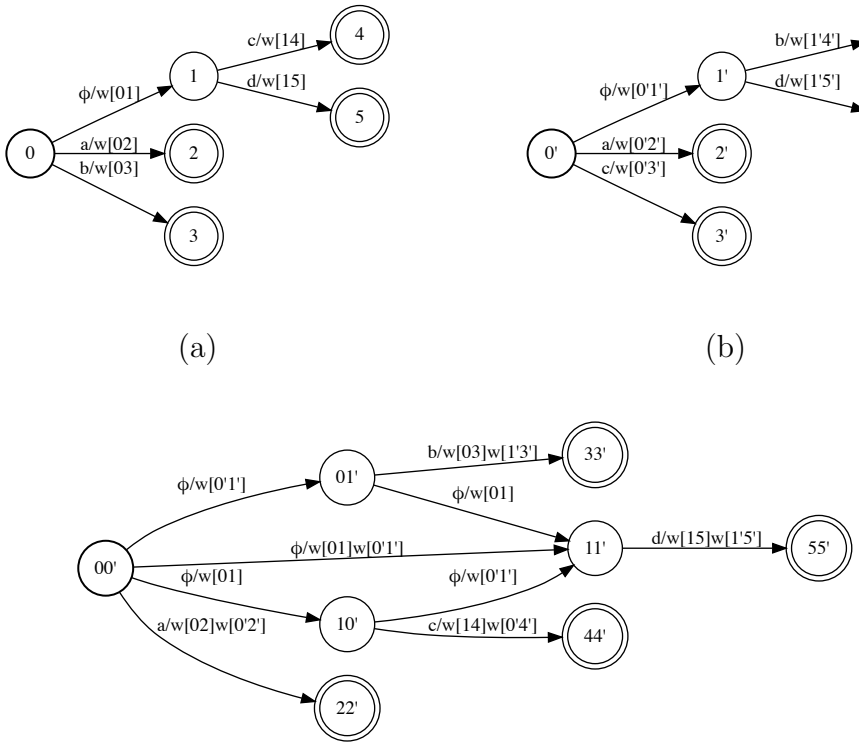


Figure 1.10: Illustration of the redundant φ-paths produced by applying a standard composition algorithm to two φ-automata. (a) and (b) are two simple φ-automata, and (c) is the result of their composition.

The reason why redundant φ-paths are generated by standard composition algorithms is similar to the reason why redundant ϵ-paths are generated during composition of automata without failure transitions [Allauzen and Mohri, 2008]. When the algorithm is in state q in WFA $\mathcal{C}$ and state $\tilde{q}$ in $\tilde{\mathcal{C}}$, both of which contain outgoing φ-transitions, the algorithm may take any of the following steps: (1) move forward on a φ-transition in $\mathcal{C}$ while staying at $\tilde{q}$; (2) move forward on a

φ-transition in $\tilde{\mathcal{C}}$ while staying in $\mathcal{C}$; or (3) move forward in both $\mathcal{C}$ and $\tilde{\mathcal{C}}$.

To design a φ-filter, we first modify the two input automata $\mathcal{C}_1$ and $\mathcal{C}_2$ to distinguish between these cases. In $\mathcal{C}_1$, for every φ-transition, we rename the label φ as φ_2. Moreover, at the source and destination states of every φ-transition, we introduce new self-loop transitions labeled with φ_1 and with weight 1. Thus, a transition labeled with φ_2 will indicate a "move forward," while a transition labeled with φ_1 will indicate a "stay." Similarly, in $\mathcal{C}_2$, we rename the φ labels as φ_1, and we introduce self-loops labeled with φ_2 and weight 1 at the source and destination states of every φ-transition. With these modifications, any φ-path resulting from the composition algorithm will include transitions of the form: (1) $(\varphi_2 : \varphi_2)$; (2) $(\varphi_1 : \varphi_1)$; or (3) $(\varphi_2 : \varphi_1)$. This observation guides us in designing a φ-filter that will eliminate redundant paths.

Theorem 1.13 (Correctness of φ-filter $\mathcal{F}$)**.** *Let $\mathcal{F}$ be the finite-state transducer illustrated in Figure 1.12. Then, $\mathcal{C}_1 \circ \mathcal{F} \circ \mathcal{C}_2$ is weighted transducer equivalent to $\tilde{C}_1 \circ \tilde{C}_2$, where $\tilde{C}_i$ is the weighted transducer obtained from C_i by "removing" φ-transitions.*

Proof. Notice first that there is at most one shortest φ-path that takes $(\varphi_2 : \varphi_1)$ transitions first. Now, we claim that if a filter disallows the following subpaths:

$$\{(\varphi_2 : \varphi_2)(\varphi_1 : \varphi_1), (\varphi_1 : \varphi_1)(\varphi_2 : \varphi_2), (\varphi_1 : \varphi_1)(\varphi_2 : \varphi_1), (\varphi_2 : \varphi_2)(\varphi_2 : \varphi_1)\},$$

then the filter will only contain φ-paths between states that are the shortest and contain $(\varphi_2 : \varphi_1)$ transitions first. To see this, suppose that the filter allowed a path that wasn't the shortest path or didn't contain $(\varphi_2 : \varphi_1)$ transitions first. If this path wasn't the shortest path but contained all $(\varphi_2 : \varphi_1)$ transitions first, then

it must contain both transitions of the form $(\varphi_2 : \varphi_2)$ and $(\varphi_1 : \varphi_1)$ after $(\varphi_2 : \varphi_1)$ transitions. But this implies that this path also contains consecutive transitions of the form $(\varphi_2 : \varphi_2)(\varphi_1 : \varphi_1)$ or $(\varphi_1 : \varphi_1)(\varphi_2 : \varphi_2)$, which are two of the disallowed subpaths. If the path didn't contain $(\varphi_2 : \varphi_1)$ transitions first, then it must begin with a transition labeled with $(\varphi_1 : \varphi_1)$ or $(\varphi_2 : \varphi_2)$. At some point, the path must include $(\varphi_2 : \varphi_1)$, which would constitute one of the disallowed subpaths.

Finally, we now show that a filter carrying the above properties can be represented as a finite-state transducer. Notice that the set of forbidden sequences can be represented by the transducer shown in Figure 1.11, and by the language:

$$L = \sigma^* \Big((\varphi_2 : \varphi_2)(\varphi_1 : \varphi_1) + (\varphi_1 : \varphi_1)(\varphi_2 : \varphi_2) + (\varphi_1 : \varphi_1)(\varphi_2 : \varphi_1) + (\varphi_2 : \varphi_2)(\varphi_2 : \varphi_1) \Big) \sigma^*,$$

where $\sigma = \{(a, a)\}_{a \in \Sigma} \cup \{(\varphi_2 : \varphi_1), (\varphi_1 : \varphi_1), (\varphi_2 : \varphi_2)\}$. Since L is a regular language, its complement, $\bar{L}$, is also a regular language and can be represented as an automaton that is the result of determinimization and complementation of an automaton accepting L. This automaton is shown in Figure 1.12. □

Figure 1.11 illustrates the automaton accepting the language L in the proof of Theorem 1.13. We define our φ-filter $\mathcal{F}$ as the finite state transducer induced by the automaton obtained after determinization and complementation of the automaton in Figure 1.11. This new FST is illustrated in Figure 1.12.

Notice that the composition of any two φ-automata and the φ-filter $\mathcal{F}$, $\mathcal{C}_1 \circ \mathcal{F} \circ \mathcal{C}_2$, will result in a finite-state transducer whose transitions have labels in $\{(a : a)\}_{a \in \Sigma} \cup \{(\varphi_2 : \varphi_2), (\varphi_1 : \varphi_1), (\varphi_2 : \varphi_1)\}$. Moreover, we identify all label pairs

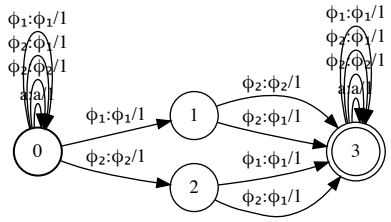


Figure 1.11: Illustration of an automaton representing the set of disallowed sequences representing language L.

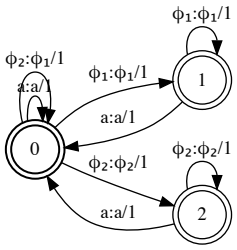


Figure 1.12: Illustration of the φ-filter $\mathcal{F}$.

in $\{(\varphi_2 : \varphi_2), (\varphi_1 : \varphi_1), (\varphi_2 : \varphi_1)\}$ using the same semantic of "other" as we did with φ. Thus, we can identify all label pairs in $\{(\varphi_2 : \varphi_2), (\varphi_1 : \varphi_1), (\varphi_2 : \varphi_1)\}$ with the single pair $(\varphi : \varphi)$ and treat the result of composition as simply a weighted finite automaton.

We are now ready to present a general φ-composition algorithm that is able to compose two φ-automata correctly. This composition algorithm will take as input three weighted finite-state transducers, two corresponding to the φ-automata, $\mathcal{C}_1$ and $\mathcal{C}_2$ and one corresponding to the φ-filter, $\mathcal{F}$, and it will output $\mathcal{C}_1 \circ \mathcal{F}\mathcal{C}_2$. A

straightforward implementation would be to recursively compose two transducers at a time (e.g. $\mathcal{C}_1 \circ \mathcal{F}\mathcal{C}_2 = (\mathcal{C}_1 \circ \mathcal{F}) \circ \mathcal{C}_2)$). However, we can improve upon this by extending the *3-way composition* technique presented in [Allauzen and Mohri, 2008] to our φ-automata setting. This algorithm, General-φ-Composition, is given as Algorithm 11. Instead of composing two automata at a time by matching transitions, this algorithm uses a *lateral strategy* to compose all three automata at once. Specifically, it first considers pairs of transitions in $\mathcal{C}_1 \times \mathcal{C}_2$, and then it searches for matching transitions in $\mathcal{F}$.

As in the original 3-way composition algorithm, we can pre-process the φ-filter $\mathcal{F}$ using perfect hashing in expected linear time $\mathcal{O}(|E_{\mathcal{F}}|)$ so that candidate transition sets $\mathcal{E}$ can be found in worst-case linear time $\mathcal{O}(|\mathcal{E}|)$. In this way, the worst-case running-time of General-φ-Composition is in

$$\mathcal{O}\left(|Q_{\mathcal{C}}| \max_{q \in \mathcal{C}_1} |E_{\mathcal{C}_1}[q]| \max_{q \in \mathcal{C}_2} |E_{\mathcal{C}_2}[q]| + |E_{\mathcal{C}}|\right).$$

The algorithms described above are *general*. While the k-shifting automaton was used as a running example, the algorithms φ-*Convert*, φ-AWM, and n-gramSelect can be applied to *any* weighted finite automaton. In practice, this automaton can even be learned from data and on-demand over a series of epochs. Given a set of experts, we can learn a language model over the sequences to determine which ones have performed well in the past. We can represent this language model using a WFA. We can then convert it into a φ-WFA or first an n-gram model and then a φ-WFA to learn against it in a computationally efficient way. As we learn using this φ-WFA, we can also simultaneously learn another language model over sequences that we can use in the next epoch.

Algorithm 11: General-φ-Composition.

Algorithm: General-φ-Composition($\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{F}$)

$Q \leftarrow I_{\mathcal{C}_1} \times I_{\mathcal{F}} \times I_{\mathcal{C}_2}$
$\mathcal{Q} \leftarrow I_{\mathcal{C}_1} \times I_{\mathcal{F}} \times I_{\mathcal{C}_2}$ **while** $\mathcal{Q} \neq \varnothing$ **do**
  $q = (q_1, q_{\mathcal{F}}, q_2) \leftarrow \text{HEAD}(\mathcal{Q})$
  DEQUEUE($\mathcal{Q}$)
  if $q \in I_{\mathcal{C}_1} \times I_{\mathcal{F}} \times I_{\mathcal{C}_2}$ **then**
    $I \leftarrow I \cup \{q\}$
    $\lambda(q) \leftarrow \lambda_{\mathcal{C}_1}(q_1)\lambda_{\mathcal{C}_2}(q_2)$
  if $q \in F_{\mathcal{C}_1} \times F_{\mathcal{F}} \times F_{\mathcal{C}_2}$ **then**
    $F \leftarrow F_{\mathcal{C}_T} \cup \{q\}$
    $\rho[q] \leftarrow \rho_{\mathcal{C}_1}[q_1]\rho_{\mathcal{C}_2}[q_2]$
  for each $(e_1, e_2) \in E_{\mathcal{C}_1} \times E_{\mathcal{C}_2}$ **do**
    if $\sigma[e_1] = \varphi$ **then**
      $\sigma[e_1] \leftarrow \varphi_2$.
      $E_{\mathcal{C}_1} \leftarrow E_{\mathcal{C}_1} \cup (q_1, \varphi_1, 1, q_1)$
      if $(\text{dest}[e_1], \varphi_1, 1, \text{dest}[e_1]) \notin E_{\mathcal{C}_1}$ **then**
        $E_{\mathcal{C}_1} \leftarrow E_{\mathcal{C}_1} \cup (\text{dest}[e_1], \varphi_1, 1, \text{dest}[e_1])$
    if $\sigma[e_2] = \varphi$ **then**
      $\sigma[e_2] \leftarrow \varphi_1$
      $E_{\mathcal{C}_2} \leftarrow E_{\mathcal{C}_2} \cup (q_2, \varphi_2, 1, q_2)$
      if $(\text{dest}[e_2], \varphi_2, 1, \text{dest}[e_2]) \notin E_{\mathcal{C}_2}$ **then**
        $E_{\mathcal{C}_2} \leftarrow E_{\mathcal{C}_2} \cup (\text{dest}[e_2], \varphi_2, 1, \text{dest}[e_2])$
    $\mathcal{E} \leftarrow \{e \in E_{\mathcal{F}} \colon i[e] = \sigma[e_1] \wedge o[e] = \sigma[e_2]\}$.
    for each $e_{\mathcal{F}} \in \mathcal{E}$ **do**
      if $\tilde{q} = (\text{dest}[e_1], \text{dest}[e_{\mathcal{F}}], \text{dest}[e_2]) \notin Q$ **then**
        $Q \leftarrow Q \cup \{\tilde{q}\}$
        ENQUEUE($\mathcal{Q}, \tilde{q}$)
      if $\sigma[e_1] \in \{\varphi_1, \varphi_2\}$ **then**
        $s \leftarrow \varphi$
      else
        $s \leftarrow \sigma[e_1]$
      $E \leftarrow E \uplus \{(q, s, w[e_1]w[e_2], \tilde{q})\}$
$\mathcal{C} \leftarrow (\Sigma, Q, I, F, E, \lambda, \rho)$
return $\mathcal{C}$

1.10 Extension to sleeping experts

In many real-world applications, it may be natural for some experts to abstain from making predictions on some of the rounds. For instance, in a bag-of-words

model for document classification, the presence of a feature or subset of features in a document can be interpreted as an expert that is awake. This extension of standard prediction with expert advice is also known as the *sleeping experts framework* [Freund et al., 1997]. The experts are said to be asleep when they are inactive and awake when they are active and available to be selected. This framework is distinct from the permutation-based definitions adopted in the studies by Kleinberg et al. [2010], Kanade et al. [2009], Kanade and Steinke [2014].

Formally, at each round t, the adversary chooses an awake set $A_t \subseteq \Sigma$ from which the learner is allowed to query an expert. The algorithm then (randomly) chooses an expert i_t from A_t, receives a loss vector $l_t \in [0,1]^{|\Sigma|}$ supported on A_t and incurs loss $l_t[i_t]$. Since some experts may not be available in some rounds, it is not reasonable to compare the loss against that of the best static expert or sequence of experts. In [Freund et al., 1997], the comparison is made against the best fixed mixture of experts normalized at each round over the awake set: $\min_{u \in \Delta_N} \sum_{t=1}^{T} \frac{\sum_{i \in A_t} u_i l_t[i]}{\sum_{j \in A_t} u_j}$.

We extend the notion of sleeping experts to the path setting, so that instead of comparing against fixed mixtures over experts, we compare against fixed mixtures over the family of expert sequences. With some abuse of notation, let A_t also represent the automaton accepting all paths of length T whose t-th transition has label in A_t. Thus, we want to design an algorithm that performs well with respect to the following quantity:

$$\min_{u \in \Delta_K} \sum_{t=1}^{T} \frac{\sum_{\pi \in \mathcal{C}_T \cap A_t} u_\pi l_t[\sigma_t[\pi]]}{\sum_{\pi \in \mathcal{C}_T \cap A_t} u_\pi}.$$

This motivates the design of AwakePBWM, a path-based weighted majority

algorithm that generalizes the algorithms in [Freund et al., 1997] to arbitrary families of expert sequences. Like PBWM, AwakePBWM maintains a set of weights over all the paths in the input automaton. At each round t, the algorithm performs a weighted majority-type update. However, it normalizes the weights so that the total weight of the awake set remains unchanged. This prevents the algorithm from "overfitting" to experts that have been asleep for many rounds. The pseudocode of this algorithm is presented as Algorithm 12, and its accompanying guarantee is given in Theorem 1.14, whose proof is in Appendix A.1.

Algorithm 12: AwakePath-BasedWeightedMajority(AwakePBWM).

Algorithm: AwakePBWM($\mathcal{C} \cap \mathcal{S}_T$, $\tilde{v}_1$),
where $\tilde{v}_1 \in \mathbb{R}_+^K$ is a vector of initial path weights.

$\boldsymbol{\pi} \leftarrow \mathcal{C} \cap \mathcal{S}_T$
$K \leftarrow |\mathcal{C} \cap \mathcal{S}_T|$
$N \leftarrow |\Sigma|$
for $j = 1$ *to* K **do**
  $\tilde{\mathsf{p}}_{1,j} \leftarrow \frac{\tilde{v}_{1,j}}{\sum_{i=1}^{K} \tilde{v}_{1,i}}$
for $j = 1$ *to* N **do**
  $\mathsf{p}_{1,j} \leftarrow \sum_{i \in [1,K]:\, \pi_{i,1}=j} \tilde{\mathsf{p}}_{1,i}$
for $t = 1$ *to* T **do**
  Receive(A_t)
  for $j = 1$ *to* N **do**
    $\mathsf{p}_{t,j}^{A_t} \leftarrow \frac{p_{t,j}}{\sum_{i \in A_t} \mathsf{p}_{t,i}}$.
  $i_t \leftarrow$Sample($\mathsf{p}_t^{A_t}$)
  Play(i_t)
  Receive($\mathbf{l}_t$)
  for $j = 1$ *to* K **do**
    $\tilde{v}_{t+1,j} \leftarrow \tilde{v}_{t,j} e^{-\eta l_t[\pi_{j,t}]}$
    $\tilde{\mathsf{p}}_{t+1,j} \leftarrow \frac{\tilde{v}_{t+1,j}}{\sum_{i=1}^{K} \tilde{v}_{t+1,i}}$
  for $j = 1$ *to* N **do**
    $\mathsf{p}_{t+1,j}^{\dagger} \leftarrow \sum_{i \in [1,K]:\, \pi_{i,t}=j} \tilde{\mathsf{p}}_{t+1,i}$
  for $j = 1$ *to* N **do**
    $\mathsf{p}_{t+1,j} \leftarrow p_{t+1,j}^{\dagger} \frac{\sum_{i \in A_t} \mathsf{p}_{t,i}}{\sum_{i \in A_t} \mathsf{p}_{t+1,i}^{\dagger}}$

Theorem 1.14 (Regret Bound for AwakePBWM)**.** *Let* $\tilde{v}_{1,j} = 1$ *for every* $j \in [K]$ *be the initial path weights in the* AwakePBWM *algorithm. For each* $t \in [T]$*, let* $A_t \subset \Sigma$ *denote the set of experts that are awake at time* t*. Then for any distribution* $u \in \Delta_K$*,* AwakePBWM *admits the following unweighted regret guarantee:*

$$\sum_{t=1}^{T} u(A_t) \mathop{\mathbb{E}}_{i \sim \mathsf{p}_t^{A_t}} [l_t[i]] - \sum_{i \in [K] \colon \pi_{i,t} \in A_t} u_i l_t[\pi_{i,t}] \leq \frac{\eta}{8} \sum_{t=1}^{T} u(A_t) + \frac{1}{\eta} \log(K),$$

where $u(A_t) = \sum_{i \in [K] \colon \pi_{i,t} \in A_t} u_i$*.*

As with PBWM, the per-iteration computational complexity of AwakePBWM is in $O(|\mathcal{C}_T|)$, the number of paths in consideration. We improve upon this by extending AWM to the sleeping expert setting and designing a new algorithm called AwakeAWM. The key modification in AwakeAWM is that during the flow computations, we intersect the set of available edges with A_t before performing the weight update. This intersection can be performed while preserving the favorable computational complexity of AWM, which depends on the states in $\mathcal{C}_T$ reachable at times t times the maximum out-degree of any state. Algorithm 13 provides the pseudocode for AwakeAWM. As in the non-sleeping expert setting, we can further improve the computational complexity by applying φ-conversion to arrive at a or n-gram approximation and then φ-conversion. All other improvements in the sleeping expert setting will similarly mirror those in the non-sleeping expert algorithms.

Algorithm 13: AwakeAutomataWeightedMajority(AwakeAWM).

Algorithm: AwakeAWM($\mathcal{C}$)

$\mathcal{C}_T \leftarrow \mathcal{C} \cap \mathcal{S}_T$
$\mathcal{C}_T \leftarrow \textsc{Connect}(\mathcal{C}_T)$
$\boldsymbol{\beta} \leftarrow \textsc{ShortestDistance}((\mathcal{C}_T)^R, F)$
$w_{\mathcal{C}_T}[I_{\mathcal{C}_T}] \leftarrow w_{\mathcal{C}_T}[I_{\mathcal{C}_T}]/\beta[I_{\mathcal{C}_T}]$
$\boldsymbol{\alpha} \leftarrow 0$
$\mathbf{Flow}_0 \leftarrow 0$
for each $e \in E_{\mathcal{C}_T}$ **with** $\mathrm{dest}[e] = (\cdot, 1)$ **do**
    $\mathrm{flow}[e] \leftarrow \beta[\mathrm{dest}[e]]$
    $\mathrm{Flow}_0[\sigma[e]] \leftarrow \mathrm{Flow}_0[\sigma[e]] + \mathrm{flow}[e]$
    $Z_0 \leftarrow Z_0 + \mathrm{flow}[e]$
$\mathsf{p}_1 \leftarrow \mathrm{Flow}_0/Z_0$
for $t \leftarrow 1$ **to** T **do**
    $\textsc{Receive}(A_t)$
    $P_t^{A_t} \leftarrow \sum_{j \in A_t} \mathsf{p}_t[j]$
    for each $i \in A_t$ **do**
        $\mathsf{p}_t^{A_t}[i] \leftarrow \mathsf{p}_t[i]/P_t^{A_t}$
    $i_t \leftarrow \textsc{Sample}(\mathsf{p}_t^{A_t})$
    $\textsc{Play}(i_t)$
    $\textsc{Receive}(\mathbf{l}_t)$
    $\mathbf{Flow}_t \leftarrow 0$
    for each $e \in E_{\mathcal{C}_T}$ **with** $\mathrm{dest}[e] = (\cdot, t)$ **do**
        if $\sigma[e] \in A_t$ **then**
            $w[e] \leftarrow w[e] \cdot e^{-\eta l_t[\sigma[e]]}$
        $\mathrm{flow}[e] \leftarrow \alpha[\mathrm{src}[e]] w[e] \beta[\mathrm{dest}[e]]$
        $\mathrm{Flow}_t[\sigma[e]] \leftarrow \mathrm{Flow}_t[\sigma[e]] + \mathrm{flow}[e]$
        $Z_t^{A_t} \leftarrow 0$
        if $\sigma[e] \in A_t$ **then**
            $Z_t^{A_t} \leftarrow Z_t^{A_t} + \mathrm{flow}[e]$
    $\mathsf{p}_{t+1} \leftarrow \mathbf{Flow}_t \frac{P_t^{A_t}}{Z_t^{A_t}}$
    $\boldsymbol{\alpha} \leftarrow \textsc{IncrSD}(\mathcal{C}_T, t, \boldsymbol{\alpha})$

1.11 Extension to online convex optimization

We now illustrate how the framework described in this paper can be extended to the general online convex optimization (OCO) setting. Online convex optimization

is a sequential prediction game over a compact convex action space $\mathcal{K}$. At each round t, the learner plays an action $x_t \in \mathcal{K}$ and receives a convex loss function f_t. The goal of the learner is to minimize the regret against the best static loss:

$$\sum_{t=1}^{T} f_t(x_t) - \min_{z \in \mathcal{K}} \sum_{t=1}^{T} f_t(z).$$

As in the framework introduced above, we can generalize this notion of regret to one against families of sequences. Specifically, let $\mathcal{C}_T \subseteq \mathcal{K}^T$ be a closed subset, let $\mathsf{p}_{\mathcal{C}_T}$ be a distribution over $\mathcal{C}_T$, and and let $\mathbf{u}_{\mathcal{C}_T}$ be the uniform distribution over $\mathcal{C}_T$. The uniform distribution is well-defined, since $\mathcal{K}$ being a compact set implies that $\mathcal{K}^T$ is compact. Then we would like to compete against the following regret against $\mathsf{p}_{\mathcal{C}_T}$:

$$\mathrm{Reg}_T(\mathcal{A}, \mathcal{C}_T) = \max_{z_1^T \in \mathcal{C}_T} \sum_{t=1}^{T} f_t(x_t) - f_t(z_t) + \log\left(\frac{\mathsf{p}_{\mathcal{C}_T}(z_1^T)}{u_{\mathcal{C}_T}}\right). \tag{1.10}$$

If $\mathsf{p}_{\mathcal{C}_T}$ is uniform, then the last term vanishes.

When $\mathcal{C}_T = \mathcal{K}^T$ is the family of all sequences of length T and $\mathsf{p}_{\mathcal{C}_T}$ is the uniform distribution, this problem has been studied in [Hall and Willett, 2013, György and Szepesvári, 2016]. In both works, the authors introduce a variant of mirror descent which applies a mapping after the standard mirror descent update, which is called DynamicMirrorDescent (DMD) in the first paper.

Specifically, if $g_t \in \partial f_t(x_t)$ is an element of the subgradient, and D_ψ is the Bregman divergence induced by a mirror map ψ, then DynamicMirrorDescent

consists of the update rule:

$$\tilde{x}_{t+1} \leftarrow \operatorname*{argmin}_{x \in \mathcal{K}} \langle g_t, x \rangle + D_\psi(x, x_t)$$
$$x_{t+1} \leftarrow \Phi_t(\tilde{x}_{t+1}).$$

In this algorithm, Φ_t is an arbitrary mapping that is specified by the learner at time t.

Under certain assumptions on the loss functions, Ψ, and Φ_t, DynamicMirrorDescent achieves the following regret guarantee against the competitor distribution $\mathsf{p}_{\mathcal{C}_T}$:

Theorem 1.15 (DynamicMirrorDescent regret against $\mathcal{C}_T$)**.** *Suppose that the Φ_ts chosen in* DynamicMirrorDescent *are non-expansive under the Bregman divergence D_ψ:*

$$D_\psi(\Phi_t(x), \Phi_t(y)) \leq D_\psi(x, y), \quad \forall x, y \in \mathcal{K}.$$

Furthermore, assume that f_t is uniformly L-Lipschitz in the norm $\|\cdot\|$ and that Ψ is 1-strongly convex in the same norm. Let $x_1 \in \mathcal{K}$ be given and define $D_{\max} = \sup_{z \in \mathcal{K}} D_\psi(z, x_1)$. Then, DynamicMirrorDescent *achieves the following regret guarantee:*

$$\operatorname{Reg}_T(\mathcal{A}, \mathcal{C}_T) \leq \frac{D_{\max}}{\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \|g_t\|^2 + \max_{z_1^T \in \mathcal{C}_T} \left\{ \log\left(\frac{\mathsf{p}_{\mathcal{C}_T}(z_1^T)}{u_{\mathcal{C}_T}} \right) + \frac{2}{\eta} \sum_{t=1}^{T} \psi(z_{t+1}) - \psi(\Phi_t(z_t)) - \langle \nabla \psi(x_{t+1}), z_{t+1} - \Phi_t(z_t) \rangle \right\}.$$

The proof of this result follows along similar lines as the original result by [Hall and Willett, 2013]. The major difference is that the authors in that work assume Ψ

to be Lipschitz. This allows them to derive a slightly weaker but more interpretable bound. However, it is also an assumption that we specifically choose to avoid, since mirror descent algorithms including the ExponentiatedGradient (EG) use mirror maps that are not Lipschitz. Hall and Willett [2013] also derive a bound for standard regret as opposed to regret against a distribution of sequences.

The first two terms in the regret bound are standard in online convex optimization, and the last term is the price of competing against arbitrary sequences. Note that György and Szepesvári [2016] present the same algorithm but with a different analysis and upper bound.

Proof. By standard properties of the Bregman divergence and convexity, we can compute

$$\begin{aligned}
\sum_{t=1}^{T} f_t(x_t) - f_t(z_t) &= \sum_{t=1}^{T} f_t(x_t) - f_t(z_t) + f_t(\tilde{x}_{t+1}) - f_t(\tilde{x}_{t+1}) \\
&\leq \frac{1}{\eta}\langle \nabla\psi(x_t) - \nabla\psi(\tilde{x}_{t+1}), \tilde{x}_{t+1} - z_t\rangle + f_t(x_t) - f_t(\tilde{x}_{t+1}) \\
&= \frac{1}{\eta}\left[D_\psi(z_t, x_t) - D_\psi(z_t, \tilde{x}_{t+1}) - D_\psi(\tilde{x}_{t+1}, x_t)\right] + f_t(x_t) - f_t(\tilde{x}_{t+1}) \\
&= \frac{1}{\eta}\Big[D_\psi(z_t, x_t) - D_\psi(z_{t+1}, x_{t+1}) + D_\psi(z_{t+1}, x_{t+1} - D_\psi(\Phi_t(z_t), x_{t+1}) \\
&\quad - D_\psi(z_t, \tilde{x}_{t+1}) + D_\psi(\Phi_t(z_t), x_{t+1}) - D_\psi(\tilde{x}_{t+1}, x_t)\Big] \\
&\quad + f_t(x_t) - f_t(\tilde{x}_{t+1}).
\end{aligned}$$

Since Φ_t is assumed to be non-expansive and $x_{t+1} = \Phi_t(x_{t+1})$, it follows that $-D_\psi(z_t, \tilde{x}_{t+1}) + D_\psi(\Phi_t(z_t), x_{t+1} \leq 0$.

Since Ψ is 1-strongly convex with respect to $\|\cdot\|$, it follows that $D_\psi(\tilde{x}_{t+1}, x_t) \geq$

$\frac{1}{2}\|\tilde{x}_{t+1} - x_t\|^2$. Thus, we can compute

$$\begin{aligned}
&-\frac{1}{\eta}D_\psi(\tilde{x}_{t+1}, x_t) + f_t(x_t) - f_t(\tilde{x}_{t+1}) \\
&\leq -\frac{1}{2\eta}\|\tilde{x}_{t+1} - x_t\|^2 + f_t(x_t) - f_t(\tilde{x}_{t+1}) \\
&\leq -\frac{1}{2\eta}\|\tilde{x}_{t+1} - x_t\|^2 + \|g_t\|_*\|x_t - \tilde{x}_{t+1}\| \\
&\leq -\frac{1}{2\eta}\|\tilde{x}_{t+1} - x_t\|^2 + \frac{\eta}{2}\|g_t\|_*^2 + \frac{1}{2\eta}\|x_t - \tilde{x}_{t+1}\|^2 = \frac{\eta}{2}\|g_t\|_*^2.
\end{aligned}$$

Moreover, we can also write

$$\begin{aligned}
&D_\psi(z_{t+1}, x_{t+1} - D_\psi(\Phi_t(z_t), x_{t+1}) \\
&= \psi(z_{t+1}) - \psi(x_{t+1}) - \langle \nabla\psi(x_{t+1}), z_{t+1} - x_{t+1}\rangle \\
&\quad [\psi(\Phi_t(z_t)) - \psi(x_{t+1}) - \langle \nabla\psi(x_{t+1}), \Phi_t(z_t) - x_{t+1}\rangle] \\
&= \psi(z_{t+1}) - \psi(\Phi_t(z_t)) - \langle \psi(x_{t+1}), z_{t+1} - \Phi_t(z_t)\rangle.
\end{aligned}$$

Combining this inequality with the inequality above yields

$$\begin{aligned}
&\sum_{t=1}^{T} f_t(x_t) - f_t(z_t) \\
&\leq \frac{1}{\eta}\sum_{t=1}^{T} D_\psi(z_t, x_t) - D_\psi(z_{t+1}, x_{t+1}) \\
&\quad + \sum_{t=1}^{T} \psi(z_{t+1}) - \psi(\Phi_t(z_t)) - \langle \psi(x_{t+1}), z_{t+1} - \Phi_t(z_t)\rangle + \frac{1}{\eta}\sum_{t=1}^{T}\frac{\eta}{2}\|g_t\|_*^2 \\
&\leq \frac{1}{\eta}D_\psi(z_1, x_1) + \sum_{t=1}^{T} \psi(z_{t+1}) - \psi(\Phi_t(z_t)) - \langle \psi(x_{t+1}), z_{t+1} - \Phi_t(z_t)\rangle \\
&\quad + \frac{1}{\eta}\sum_{t=1}^{T}\frac{\eta}{2}\|g_t\|_*^2.
\end{aligned}$$

Adding in $\log\left(\frac{\mathsf{p}_{\mathcal{C}_T}(z_1^T)}{u_{\mathcal{C}_T}}\right)$ to both sides and taking the max over $z_1^T \in \mathcal{C}_T$ completes the proof. $\square$

By restricting our competitor set to $\mathcal{C}_T$ and adding the penalization term, it follows that DynamicMirrorDescent achieves the following guarantee:

$$\begin{aligned}
&\max_{z_1^T \in \mathcal{C}_T} \sum_{t=1}^{T} f_t(x_t) - f_t(z_t) + \log\left(\frac{\mathsf{p}_{\mathcal{C}_T}(z_1^T)}{u_{\mathcal{C}_T}}\right) \\
&\quad \leq \frac{D_{\max}}{\eta} + \frac{\eta}{2}\sum_{t=1}^{T} \|g_t\|^2 + \max_{z_1^T \in \mathcal{C}_T} \left\{ \log\left(\frac{\mathsf{p}_{\mathcal{C}_T}(z_1^T)}{u_{\mathcal{C}_T}}\right) \right. \\
&\qquad \left. + \frac{2}{\eta}\sum_{t=1}^{T} \psi(z_{t+1}) - \psi(\Phi_t(z_t)) - \langle \nabla\psi(x_{t+1}), z_{t+1} - \Phi_t(z_t)\rangle \right\}.
\end{aligned}$$

This bound suggests that if we are able to find a sequence $(\Phi_t)_{t=1}^T$ that minimizes the last quantity, then we can tightly bound our regret. Now let $\mathcal{F}$ be a family of dynamic maps Φ that are non-expansive with respect to D_ψ. Then we want to solve the following optimization problem:

$$\begin{aligned}
&\min_{\Phi_1^T \in \mathcal{F}^T} \max_{z_1^T \in \mathcal{C}_T} \left\{ \log\left(\frac{\mathsf{p}_{\mathcal{C}_T}(z_1^T)}{u_{\mathcal{C}_T}}\right) \right. \\
&\quad \left. + \frac{2}{\eta}\sum_{t=1}^{T} \psi(z_{t+1}) - \psi(\Phi_t(z_t)) - \langle \nabla\psi(x_{t+1}), z_{t+1} - \Phi_t(z_t)\rangle \right\}. \qquad (1.11)
\end{aligned}$$

We can view this as the online convex optimization analogue of the automata approximation problem in Section 1.6, and we can use it in the same way to derive concrete online convex optimization algorithms that achieve good regret against more complex families of sequences.

As an illustrative example, we apply this to the k-shifting experts setting and show how a candidate solution to this problem recovers the Fixed-Share

algorithm.

1.11.1 OCO derivation of Fixed-Share

Suppose that we are again in the prediction with expert advice setting so that $\mathcal{K} = \Delta_N$ and $f_t(x) = \langle l_t, x\rangle$. Assume that $\mathcal{C}_T$ is the set of k-shifting experts and that $\mathsf{p}_{\mathcal{C}_T}$ is the uniform distribution on $\mathcal{C}_T$. As for the weighted majority algorithm, let $\Psi = \sum_{i=1}^N x_i \log(x_i)$ be the negative entropy so that $D_\psi(x,y) = \sum_{i=1}^N x_i \log\left(\frac{x_i}{y_i}\right)$ is the relative entropy. One way of ensuring that Φ_t is non-expansive is to define it to be a mixture with a fixed vector: $\Phi_t(x) = (1-\alpha_t)x + \alpha_t w_t$ for some $w_t \in \Delta_N$ and $\alpha_t \in [0,1]$. By convexity of the relative entropy, it follows that for any $x, y \in \Delta_N$, $D_\psi(\Phi_t(x), \Phi_t(y)) \leq D_\psi(x,y)$.

For simplicity, we can assume that $\Phi_t = \Phi$. Then Problem 1.11 can be written as:

$$\min_{w\in\Delta_N, \alpha\in[0,1]} \max_{z_1^T \in \mathcal{C}_T} \left\{ \frac{2}{\eta} \sum_{t=1}^T \psi(z_{t+1}) - \psi((1-\alpha)z_t + \alpha w) \right.$$
$$\left. - \langle \nabla\psi((1-\alpha)\tilde{x}_{t+1} + \alpha w), z_{t+1} - (1-\alpha)z_t - \alpha w\rangle \right\}.$$

Since $\mathcal{C}_T$ is symmetric across coordinates and we do not have a priori knowledge of of $\tilde{x}_{t+1}$, a reasonable choice of w is the uniform distribution $w_i = \frac{1}{N}$. We can also use the fact that the entropy function is convex to obtain the upper bound: $-\psi((1-\alpha)z_t + \alpha w) \leq -(1-\alpha)\psi(z) - \alpha\psi(w)$. Moreover, since z_t is always only supported on a single coordinate, $\psi(z_t) = 0$ for every t.

This reduces to the following optimization problem:

$$\min_{\alpha\in[0,1]} \max_{z_1^T\in\mathcal{C}_T} \Bigg\{ \frac{2}{\eta}\sum_{t=1}^{T} \alpha\log(N) \\ - \sum_{i=1}^{N} \log\left((1-\alpha)\tilde{x}_{t+1,i} + \alpha\frac{1}{N}\right)\left[z_{t+1,i} - (1-\alpha)z_{t,i} - \alpha\frac{1}{N}\right]\Bigg\}.$$

We can break the objective into three separate terms:

$$A_1 : \frac{2}{\eta}\sum_{t=1}^{T}\alpha\log(N)$$
$$A_2 : -\sum_{t=1}^{T}\sum_{i=1}^{N}\log\left((1-\alpha)\tilde{x}_{t+1,i} + \alpha\frac{1}{N}\right)[z_{t+1,i} - z_{t,i}]$$
$$A_3 : -\sum_{t=1}^{T}\sum_{i=1}^{N}\log\left((1-\alpha)\tilde{x}_{t+1,i} + \alpha\frac{1}{N}\right)\alpha\left[z_{t,i} - \frac{1}{N}\right]$$

It is straightforward to see that $A_1 = \frac{2}{\eta}T\alpha\log(N)$. To bound A_2, let $i_t \in [N]$ be the index such that $z_{t,i_t} = 1$ and $z_{t,i} = 0$ for all $i \neq i_t$. Then,

$$-\sum_{t=1}^{T}\sum_{i=1}^{N}\log\left((1-\alpha)\tilde{x}_{t+1,i} + \alpha\frac{1}{N}\right)[z_{t+1,i} - z_{t,i}]$$
$$= -\sum_{t:i_{t+1}\neq i_t}\sum_{i=1}^{N}\log\left((1-\alpha)\tilde{x}_{t+1,i} + \alpha\frac{1}{N}\right)[z_{t+1,i} - z_{t,i}]$$
$$\leq -\sum_{t:i_{t+1}\neq i_t}\sum_{i=1}^{N}\log\left(\alpha\frac{1}{N}\right)z_{t+1,i}$$
$$\leq -k\log\left(\frac{\alpha}{N}\right).$$

To bound A_3, we can write

$$\begin{aligned}
&-\alpha \sum_{t=1}^{T} \sum_{i=1}^{N} \log\left((1-\alpha)\tilde{x}_{t+1,i} + \alpha\frac{1}{N}\right)\left[z_{t,i} - \frac{1}{N}\right] \\
&= -\alpha \sum_{t=1}^{T} \log\left((1-\alpha)\tilde{x}_{t+1,i_t} + \alpha\frac{1}{N}\right)\left[1 - \frac{1}{N}\right] \\
&\leq -\alpha T \log\left(\alpha\frac{1}{N}\right).
\end{aligned}$$

Putting the pieces together, the objective is bounded by

$$\frac{2}{\eta}\left(T\alpha\log(N) - k\log\left(\frac{\alpha}{N}\right) - \alpha T\log\left(\frac{\alpha}{N}\right)\right),$$

leading to the new optimization problem:

$$\min_{\alpha\in[0,1]} \frac{2}{\eta}\left(T\alpha\log(N) - k\log\left(\frac{\alpha}{N}\right) - \alpha T\log\left(\frac{\alpha}{N}\right)\right).$$

Notice that $\alpha \propto \frac{k}{T}$ is a reasonable solution, as it bounds the regret by $\mathcal{O}\left(k\log\left(\frac{NT}{k}\right)\right)$.

Moreover, this choice of α approximately corresponds to Fixed-Share. Thus, we have again derived the Fixed-Share algorithm from first principles in consideration of only the k-shifting expert sequences. This is in contrast to previous work for DynamicMirrorDescent (e.g. [György and Szepesvári, 2016]) which only showed that one could define Φ_t in a way that mimics the Fixed-Share algorithm.

1.12 Summary of chapter

We studied a general framework of online learning against a competitor class represented by a weighted automaton and showed that sublinear regret guarantees

can be achieved in this scenario.

We gave a series of algorithms for this problem, including an automata-based algorithm extending weighted-majority whose computational cost at round t depends on the total number of transitions leaving the states of the competitor automaton reachable at time t, which substantially improves upon a naïve algorithm based on path updates. We used the notion of failure transitions to provide a compact representation of the competitor automaton or its intersection with the set of strings of length t, thereby resulting in significant efficiency improvements. This required the introduction of new failure-transition-based composition and shortest-distance algorithms that could be of independent interest.

We further gave an extensive study of algorithms based on a compact approximation of the competitor automata. We showed that the key quantity arising when using an approximate weighted automaton is the Rényi divergence of the original and approximate automata. We presented a specific study of approximations based on n-gram models by minimizing the Rényi divergence and studied the properties of maximum likelihood n-gram models. We pointed out the efficiency benefits of such approximations and provides guarantees on the approximations and the regret. We also extended our algorithms and results to the framework of sleeping experts. We further described the extension of the approximation methods to online convex optimization and a general mirror descent setting.

Our description of this general (weighted) regret minimization framework and the design of algorithms based on automata provides a unifying view of many similar problems and leads to general algorithmic solutions applicable to a wide variety of problems with different competitor class automata. In general, automata lead to a more general and cleaner analysis. An alternative approximation method

consists of directly minimizing the competitor class automaton before intersection with the set of strings of length t. We have also studied that method, presented guarantees for its success, and illustrated the approach in a special case.

Note that, instead of automata and regular languages, we could have considered more complex formal language families such as (probabilistic) context-free languages over expert sequences. However, more complex languages can be handled in a similar way since the intersection with $\mathcal{S}_T$ would be a finite language. The method based on a direct approximation would require approximating a probabilistic context-free language using weighted automata, a problem that has been extensively studied in the past [Pereira and Wright, 1991, Nederhof, 2000, Mohri and Nederhof, 2001].

Finally, all our results can be straightforwardly extended to the adversarial bandit scenario using standard surrogate losses based on importance weighting techniques.

Chapter 2

Conditional Swap Regret and Conditional Correlated Equilibrium

In this chapter, we introduce a natural extension of the notion of swap regret, *conditional swap regret*, that allows for action modifications conditioned on the player's action history. We prove a series of new results for conditional swap regret minimization. We present algorithms for minimizing conditional swap regret with bounded conditioning history. We further extend these results to the case where conditional swaps are considered only for a subset of actions. We also define a new notion of equilibrium, *conditional correlated equilibrium*, that is tightly connected to the notion of conditional swap regret: when all players follow conditional swap regret minimization strategies, then the empirical distribution approaches this equilibrium. Finally, we extend our results to the multi-armed bandit scenario.

The standard measure of the quality of an online algorithm is its *regret*, which

is the difference between the cumulative loss it incurs after some number of rounds and that of an alternative policy. The cumulative loss can be compared to that of the single best action in retrospect Littlestone and Warmuth [1994] (*external regret*), to the loss incurred by changing every occurrence of a specific action to another Foster and Vohra [1997] (*internal regret*), or, more generally, to the loss of action sequences obtained by mapping each action to some other action Blum and Mansour [2007] (*swap regret*). Swap regret, in particular, accounts for situations where the algorithm could have reduced its loss by swapping every instance of one action with another (e.g. every time the player bought Microsoft, he should have bought IBM).

There are many algorithms for minimizing external regret [Cesa-Bianchi and Lugosi, 2006], such as, for example, the randomized weighted-majority algorithm of Littlestone and Warmuth [1994]. It was also shown in Blum and Mansour [2007] and Stoltz and Lugosi [2007] that there exist algorithms for minimizing internal and swap regret. These regret minimization techniques have been shown to be useful for approximating game-theoretic equilibria: external regret algorithms for Nash equilibria and swap regret algorithms for correlated equilibria Nisan et al. [2007].

By definition, swap regret compares a player's action sequence against all possible modifications at each round, independently of the previous time steps. In this paper, we introduce a natural extension of swap regret, *conditional swap regret*, that allows for action modifications conditioned on the player's action history. Our definition depends on the number of past time steps we condition upon.

As a motivating example, let us limit this history to just the previous one time step, and suppose we design an online algorithm for the purpose of investing, where one of our actions is to buy bonds and another to buy stocks. Since bond and stock

prices are known to be negatively correlated, we should always be wary of buying one immediately after the other – unless our objective was to pay for transaction costs without actually modifying our portfolio! However, this does not mean that we should avoid purchasing one or both of the two assets completely, which would be the only available alternative in the swap regret scenario. The conditional swap class we introduce provides precisely a way to account for such correlations between actions. We start by introducing the learning set-up and the key notions relevant to our analysis (Section 2.1).

2.1 Preliminaries

We consider the standard online learning set-up with a set of actions $\Sigma = \{1, \dots, N\}$. At each round $t \in \{1, \dots, T\}$, $T \geq 1$, the player selects an action $x_t \in \Sigma$ according to a distribution p_t over Σ, in response to which the adversary chooses a function $f_t \colon \Sigma^t \to [0, 1]$ and causes the player to incur a loss $f_t(x_t, x_{t-1}, \dots, x_1)$. The objective of the player is to choose a sequence of actions $(x_1, \dots, x_T)$ that minimizes his cumulative loss $\sum_{t=1}^{T} f_t(x_t, x_{t-1}, \dots, x_1)$.

A standard metric used to measure the performance of an online algorithm $\mathcal{A}$ over T rounds is its *(expected) external regret*, which measures the player's expected performance against the best fixed action in hindsight:

$$\mathrm{Reg}_{\mathrm{Ext}}(\mathcal{A}, T) = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\substack{(x_t, \dots, x_1) \sim \\ (\mathsf{p}_t, \dots, \mathsf{p}_1)}} [f_t(x_t, \dots, x_1)] - \min_{j \in \Sigma} \sum_{t=1}^{T} f_t(j, j, \dots, j).$$

There are several common modifications to the above online learning scenario: (1) we may compare regret against stronger competitor classes: $\mathrm{Reg}_{\mathcal{C}}(\mathcal{A}, T) =$

$\sum_{t=1}^{T} \mathbb{E}_{\mathsf{p}_t,\ldots,\mathsf{p}_1} f_t(x_t,\ldots,x_1) - \min_{\varphi\in\mathcal{C}} \sum_{t=1}^{T} \mathbb{E}_{\mathsf{p}_t,\ldots,\mathsf{p}_1}[f_t(\varphi(x_t),\varphi(x_{t-1}),\ldots,\varphi(x_1))]$ for some function class $\mathcal{C} \subseteq \Sigma^{\Sigma}$; (2) the player may have access to only partial information about the loss, i.e. only knowledge of $f_t(x_t,\ldots,x_1)$ as opposed to $f_t(a,x_{t-1},\ldots,x_1)\forall a \in \Sigma$ (also known as the *bandit scenario*); (3) the loss function may have bounded memory:

$$f_t(x_t,\ldots,x_{t-k},x_{t-k-1},\ldots,x_1) = f_t(x_t,\ldots,x_{t-k},y_{t-k-1},\ldots,y_1),$$

for any $x_j, y_j \in \Sigma$.

The scenario where $\mathcal{C} = \Sigma^{\Sigma}$ in (1) is called the *swap regret* case, and the case where $k = 0$ in (3) is referred to as the *oblivious adversary*. (Sublinear) regret minimization is possible for loss functions against any competitor class of the form described in (1), with only partial information, and with at least some level of bounded memory. See Blum and Mansour [2007] and Arora et al. [2012] for a reference on (1), Auer et al. [2002] and Bubeck et al. [2012] for (2), and Arora et al. [2012] for (3). Cesa-Bianchi et al. [2013] also provides a detailed summary of the best known regret bounds in all of these scenarios and more.

The introduction of adversaries with bounded memory naturally leads to an interesting question: *what if we also try to increase the power of the competitor class in this way?*

While swap regret is a natural competitor class and has many useful game theoretic consequences (see Nisan et al. [2007]), one important missing ingredient is that the competitor class of functions does not have memory. In fact, in most if not all online learning scenarios and regret minimization algorithms considered so far, the point of comparison has been against modification of the player's actions

at each point of time independently of the previous actions. But, as we discussed above in the financial markets example, there exist cases where a player should be measured against alternatives that depend on the past and the player should take into account the correlations between actions.

Specifically, we consider competitor functions of the form $\Phi_t \colon \Sigma^t \to \Sigma^t$. Let $\mathcal{C}_{\text{all}} = \{\Phi_t \colon \Sigma^t \to \Sigma^t\}_{t=1}^{\infty}$ denote the class of all such functions. This leads us to the expression: $\sum_{t=1}^{T} \mathbb{E}_{\mathsf{p}_t}[f_t(x_t)] - \min_{\Phi_t \in \mathcal{C}_{\text{all}}} \sum_{t=1}^{T} \mathbb{E}_{\mathsf{p}_t, dots, \mathsf{p}_1}[f_t \circ \Phi_t(x_t, \ldots, x_1)]$. $\mathcal{C}_{\text{all}}$ is clearly a substantially richer class of competitor functions than traditional swap regret. In fact, it is the most comprehensive class, since we can always reach $\sum_{t=1}^{T} \mathbb{E}_{\mathsf{p}_t}[f_t(x_t)] - \sum_{t=1}^{T} \min_{(x_t, \ldots, x_1)} f_t(x_t, \ldots, x_1)$ by choosing Φ_t to map all points to $\operatorname{argmin}_{(x_t, \ldots, x_1)} f_t(x_t, \ldots, x_1)$. Not surprisingly, however, it is not possible to obtain a sublinear regret bound against this general class.

Theorem 2.1. *No algorithm can achieve sublinear regret against the class $\mathcal{C}_{\text{all}}$, regardless of the loss function's memory.*

This result is well-known in the online learning community, but, for completeness, we include a proof in Appendix B.1. Theorem 2.1 suggests examining more reasonable subclasses of $\mathcal{C}_{\text{all}}$. To simplify the notation and proofs that follow in the paper, we will henceforth restrict ourselves to the scenario of an oblivious adversary, as in the original study of swap regret Blum and Mansour [2007]. However, an application of the batching technique of Arora et al. [2012] should produce analogous results in the non-oblivious case for all of the theorems that we provide.

Now consider the collection of competitor functions $\mathcal{C}_k = \{\varphi \colon \Sigma^k \to \Sigma\}$. Then, a player who has played actions $\{a_s\}_{s=1}^{t-1}$ in the past should have his performance compared against $\varphi(a_t, a_{t-1}, a_{t-2}, \ldots, a_{t-(k-1)})$ at time t, where $\varphi \in \mathcal{C}_k$. We call this class $\mathcal{C}_k$ of functions the *k-gram conditional swap regret class*, which also leads

us to the regret definition:

$$\mathrm{Reg}_T(\mathcal{C}_k, \mathcal{A}) = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{x_t \sim \mathsf{p}_t}[f_t(x_t)] - \min_{\varphi \in \mathcal{C}_k} \sum_{t=1}^{T} \mathop{\mathbb{E}}_{x_t \sim \mathsf{p}_t}[f_t(\varphi(x_t, a_{t-1}, a_{t-2}, \ldots, a_{t-(k-1)}))].$$

Note that this is a direct extension of swap regret to the scenario where we allow for swaps conditioned on the history of the previous $(k-1)$ actions. For $k = 1$, this precisely coincides with swap regret.

One important remark about the k-gram conditional swap regret is that it is a random quantity that depends on the particular sequence of actions played. A natural deterministic alternative would be of the form:

$$\sum_{t=1}^{T} \mathop{\mathbb{E}}_{x_t \sim \mathsf{p}_t}[f_t(x_t)] - \min_{\varphi \in \mathcal{C}_k} \sum_{t=1}^{T} \mathop{\mathbb{E}}_{(x_t, \ldots, x_1) \sim (\mathsf{p}_t, \ldots, \mathsf{p}_1)}[f_t(\varphi(x_t, x_{t-1}, x_{t-2}, \ldots, x_{t-(k-1)}))].$$

However, by taking the expectation of $\mathrm{Reg}_{\mathcal{C}_k}(\mathcal{A}, T)$ with respect to $a_{T-1}, a_{T_2}, \ldots, a_1$ and applying Jensen's inequality, we obtain

$$\begin{aligned}
&\mathrm{Reg}_T(\mathcal{C}_k, \mathcal{A}) \\
&\quad \geq \sum_{t=1}^{T} \mathop{\mathbb{E}}_{x_t \sim \mathsf{p}_t}[f_t(x_t)] - \min_{\varphi \in \mathcal{C}_k} \sum_{t=1}^{T} \mathop{\mathbb{E}}_{(x_t, \ldots, x_1) \sim (\mathsf{p}_t, \ldots, \mathsf{p}_1)}[f_t(\varphi(x_t, x_{t-1}, x_{t-2}, \ldots, x_{t-(k-1)}))],
\end{aligned}$$

and so no generality is lost by considering the randomized sequence of actions in our regret term.

Another interpretation of the bigram conditional swap class is in the context of finite-state transducers. Taking a player's sequence of actions $(x_1, \ldots, x_T)$, we may view each competitor function in the conditional swap class as an application of a finite-state transducer with N states, as illustrated by Figure 2.1. Each state encodes the history of actions $(x_{t-1}, \ldots, x_{t-(k-1)})$ and admits N outgoing transitions

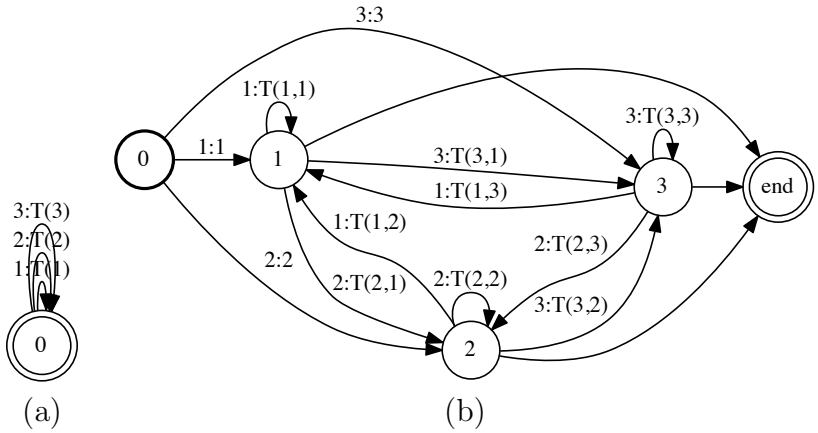


Figure 2.1: (a) unigram conditional swap class interpreted as a finite-state transducer. This is the same as the usual swap class and has only the trivial state; (b) bigram conditional swap class interpreted as a finite-state transducer. The action at time $t-1$ defines the current state and influences the potential swap at time t.

representing the next action along with its possible modification. In this framework, the original swap regret class is simply a transducer with a single state.

2.2 Full Information Scenario

Here, we prove that it is in fact possible to minimize k-gram conditional swap regret against an oblivious adversary, starting with the easier to interpret bigram scenario. We design an algorithm, BigramCSwap, that achieves sublinear bigram conditional swap regret.

BigramCSwap is a meta-algorithm that uses external regret minimizing algorithms as subroutines, as in Blum and Mansour [2007]. The key is to attribute a fraction of the loss to each external regret minimizing subroutine, so that these

Algorithm 14: BIGRAMCSWAP.

Algorithm: BIGRAMCSWAP($(\mathcal{A}_{i,j})_{i,j=1}^{N}$),
$(\mathcal{A}_{i,j})_{i,j=1}^{N}$ external-regret minimizing online learning algorithms.
$\mathsf{p}_1 \leftarrow \frac{1}{N}$.
$a_0 \leftarrow \textsc{Sample}(\mathsf{p}_1)$.
for $t \leftarrow 1, \dots, T$ **do**
  $a_t \leftarrow \textsc{Sample}(\mathsf{p}_t)$.
  $f_t \leftarrow \textsc{Play}(a_t)$.
  for $i, j \leftarrow 1, \dots, N$ **do**
    $\textsc{AssignLoss}(\mathsf{p}_t[i]\delta_{a_{t-1}}^{t-1}[j]f_t, \mathcal{A}_{i,j})$.
    $Q_i^{t+1,j} \leftarrow \textsc{AlgoStrategy}(\mathcal{A}_{i,j})$.
  $Q^{t+1} \leftarrow \sum_{j=1}^{N} \delta_{a_{t-1}}^{t-1}[j] Q^{t+1,j}$.
  $\mathsf{p}_{t+1} \leftarrow \textsc{StationaryDistribution}(Q^{t+1})$.

losses sum up to our actual realized loss. These losses will force the subroutines to minimize regret against each of the conditional swaps. The distributions provided by the subroutines can then be combined to form a Markov chain, whose stationary distribution will be the learner's strategy at each round.

Theorem 2.2. *The bigram conditional swap regret of* BIGRAMCSWAP *is bounded as follows:*

$$\mathrm{Reg}_T(\mathcal{C}_2, \textsc{BigramCSwap}) \leq \mathcal{O}\big(N\sqrt{T \log N}\big).$$

Proof. Since the distribution p_t at round t is finite-dimensional, we can represent it as a vector $\mathsf{p}_t = (\mathsf{p}_t[1], \dots, \mathsf{p}_t[N])$. Similarly, since oblivious adversaries take only N arguments, we can write f_t as the loss vector $f_t = (f_t[1], \dots, f_t[N])$. Let $\{a_t\}_{t=1}^{T}$ be a sequence of random variables denoting the player's actions at each time t, and let $\delta_{a_t}^{t}$ denote the (random) Dirac delta distribution concentrated at $a_t \in \Sigma$. Then,

we can rewrite the bigram swap regret as follows:

$$\begin{aligned}\mathrm{Reg}_T(\mathcal{C}_2, \textsc{BigramCSwap}) &= \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\mathsf{p}_t}[f_t(x_t)] - \min_{\varphi \in \mathcal{C}_2} \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\mathsf{p}_t, \delta_{a_{t-1}}^{t-1}} [f_t(\varphi(x_t, x_{t-1})] \\ &= \sum_{t=1}^{T} \sum_{i=1}^{N} \mathsf{p}_t[i] f_t[i] - \min_{\varphi \in \mathcal{C}_2} \sum_{t=1}^{T} \sum_{i,j=1}^{N} \mathsf{p}_t[i] \delta_{a_{t-1}}^{t-1}[j] f_t[\varphi(i,j)]\end{aligned}$$

Recall that an optimal external regret minimizing algorithm $\mathcal{A}_{i,j}$ (e.g. randomized weighted majority) admits an external regret bound of the form $R_{i,j} = R_{i,j}(L_{\min}^{i,j}, N) = \mathcal{O}\left(\sqrt{L_{\min}^{i,j} \log(N)}\right)$, where $L_{\min}^{i,j} = \min_{l=1}^{N} \sum_{t=1}^{T} f_t^{i,j}[l]$ for the sequence of loss vectors $(f_t^{i,j})_{t=1}^{T}$ incurred by the algorithm. Since $\mathsf{p}_t = \mathsf{p}_t Q^t$ is a stationary distribution, we can write:

$$\begin{aligned}\sum_{t=1}^{T} \mathsf{p}_t \cdot f_t &= \sum_{t=1}^{T} \sum_{l=1}^{N} \mathsf{p}_t[l] f_t[l] \\ &= \sum_{t=1}^{T} \sum_{l=1}^{N} \sum_{i=1}^{N} \mathsf{p}_t[i] Q_{i,l}^t f_t[l] \\ &= \sum_{t=1}^{T} \sum_{l=1}^{N} \sum_{i=1}^{N} \mathsf{p}_t[i] \sum_{j=1}^{N} \delta_{a_{t-1}}^{t-1}[j] Q_{i,l}^{t,j} f_t[l].\end{aligned}$$

Rearranging leads to

$$\begin{aligned}\sum_{t=1}^{T} \mathsf{p}_t \cdot f_t &= \sum_{i,j=1}^{N} \sum_{t=1}^{T} \sum_{l=1}^{N} \mathsf{p}_t[i] \delta_{a_{t-1}}^{t-1}[j] Q_{i,l}^{t,j} f_t[l] \\ &\leq \sum_{i,j=1}^{N} \left(\left(\sum_{t=1}^{T} \mathsf{p}_t[i] \delta_{a_{t-1}}^{t-1}[j] f_t[\varphi(i,j)] \right) + R_{i,j}(L_{min}^{i,j}, N) \right) \\ &\quad \text{(for arbitrary } \varphi \colon \Sigma^2 \to \Sigma) \\ &= \sum_{i,j=1}^{N} \left(\sum_{t=1}^{T} \mathsf{p}_t[i] \delta_{a_{t-1}}^{t-1}[j] f_t[\varphi(i,j)] \right) + \sum_{i,j=1}^{N} R_{i,j}(L_{min}^{i,j}, N).\end{aligned}$$

Since φ is arbitrary, we obtain

$$\begin{aligned}\mathrm{Reg}_T(\mathcal{C}_2, \textsc{BigramCSwap}) &= \sum_{t=1}^{T} \mathsf{p}_t \cdot f_t - \min_{\varphi \in \mathcal{C}_2} \sum_{t=1}^{T} \sum_{i,j=1}^{N} \mathsf{p}_t[i] \delta_{a_{t-1}}^{t-1}[j] f_t[\varphi(i,j)] \\ &\leq \sum_{i,j=1}^{N} R_{i,j}(L_{min}^{i,j}, N).\end{aligned}$$

Using the fact that $R_{i,j} = \mathcal{O}\left(\sqrt{L_{\min}^{i,j} \log(N)}\right)$ and that we scaled the losses to algorithm $\mathcal{A}_{i,j}$ by $\mathsf{p}_t[i]\delta_{a_{t-1}}^{t-1}[j]$, the following inequality holds: $\sum_{i=1}^{N}\sum_{j=1}^{N} L_{\min}^{i,j} \leq T$. By Jensen's inequality, this implies

$$\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sqrt{L_{\min}^{i,j}} \leq \sqrt{\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} L_{\min}^{i,j}} \leq \frac{\sqrt{T}}{N},$$

or, equivalently, $\sum_{i=1}^{N}\sum_{j=1}^{N}\sqrt{L_{\min}^{i,j}} \leq N\sqrt{T}$. Combining this with our regret bound yields

$$\begin{aligned}\mathrm{Reg}_T(\mathcal{C}_2, \textsc{BigramCSwap}) &\leq \sum_{i,j=1}^{N} R_{i,j}(L_{\min}, N) \\ &= \sum_{i,j=1}^{N} \mathcal{O}\left(\sqrt{L_{\min}^{i,j} \log N}\right) \leq \mathcal{O}\left(N\sqrt{T \log N}\right),\end{aligned}$$

which concludes the proof. □

Remark. The computational complexity of a standard external regret minimization algorithm such as randomized weighted majority per round is in $\mathcal{O}(N)$ (update the distribution on each of the N actions multiplicatively and then renormalize), which implies that updating the N^2 subroutines will cost $\mathcal{O}(N^3)$ per round. Allocating losses to these subroutines and combining the distributions that they return will cost

an additional $\mathcal{O}(N^3)$ time. Finding the stationary distribution of a stochastic matrix can be done via matrix inversion in $\mathcal{O}(N^3)$ time. Thus, the total computational complexity of achieving $\mathcal{O}(N\sqrt{T\log(N)})$ regret is only $\mathcal{O}(N^3T)$. We remark that in practice, one often uses iterative methods to compute dominant eigenvalues (see Trefethen and Bau [1997] for a standard reference and Halko et al. [2011] for recent improvements). Greenwald et al. [2008] has also studied techniques to avoid computing the exact stationary distribution at every iteration step for similar types of problems.

The meta-algorithm above can be interpreted in three equivalent ways: (1) the player draws an action x_t from distribution p_t at time t; (2) the player uses distribution p_t to choose among the N subsets of algorithms $Q_1^t, \ldots, Q_N^t$, picking one subset Q_i^t; next, after drawing i from p_t, the player uses $\delta_{a_{t-1}}^{t-1}$ to choose among the algorithms $Q_i^{t,1}, \ldots, Q_i^{t,N}$, picking algorithm $Q_i^{t,a_{t-1}}$; after locating this algorithm, the player uses the distribution from algorithm $Q_i^{t,a_{t-1}}$ to draw an action; (3) the player chooses algorithm $Q_i^{t,j}$ with probability $\mathsf{p}_t[i]\delta_{a_{t-1}}^{t-1}[j]$ and draws an action from its distribution.

The following more general bound can be given for an arbitrary k-gram swap scenario.

Theorem 2.3. *There exists an online algorithm $\mathcal{A}$ with k-gram swap regret bounded as follows:*

$$\mathrm{Reg}_T(\mathcal{C}_k, \mathcal{A}) \leq \mathcal{O}\big(\sqrt{N^k T \log N}\big).$$

The algorithm used to derive this result is a straightforward extension of the algorithm provided in the bigram scenario, and the proof is given in Appendix B.3.

Remark. The computational complexity of achieving the above regret bound is

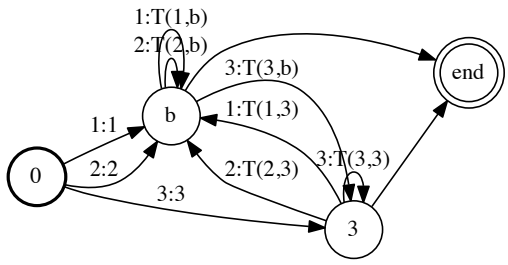


Figure 2.2: bigram conditional swap class restricted to a finite number of active states. When the action at time $t-1$ is 1 or 2, the transducer is in the same state, and the swap function is the same.

$\mathcal{O}(N^{k+1}T)$.

2.3 State-Dependent Bounds

In some situations, it may not be relevant to consider conditional swaps for every possible action, either because of the specific problem at hand or simply for the sake of computational efficiency. Thus, for any $\mathcal{S} \subseteq \Sigma^2$, we define the following competitor class of functions:

$$\mathcal{C}_{2,\mathcal{S}} = \{\varphi \colon \Sigma^2 \to \Sigma | \varphi(i,k) = \tilde{\varphi}(i) \text{ for } (i,k) \in \mathcal{S} \text{ where } \tilde{\varphi} \colon \Sigma \to \Sigma\}.$$

See Figure 2.2 for a transducer interpretation of this scenario.

We will now show that the algorithm above can be easily modified to derive a tighter bound that is dependent on the number of states in our competitor class. We will focus on the bigram case, although a similar result can be shown for the general k-gram conditional swap regret.

Theorem 2.4. *There exists an online algorithm $\mathcal{A}$ such that*

$$\mathrm{Reg}_T(\mathcal{C}_{2,\mathcal{S}}, \mathcal{A}) \leq \mathcal{O}(\sqrt{T(|\mathcal{S}^c| + N)\log(N)}).$$

The proof of this result is given in Appendix B.2. Note that when $\mathcal{S} =$, we are in the scenario where all the previous states matter, and our bound coincides with that of the previous section.

Remark. The computational complexity of achieving the above regret bound is $\mathcal{O}((N(|\pi_1(\mathcal{S})|+|\mathcal{S}^c|)+N^3)T)$, where π_1 is projection onto the first component. This follows from the fact that we allocate the same loss to all $\{\mathcal{A}_{i,k}\}_{k:(i,k)\in\mathcal{S}}$ $\forall i \in \pi_1(\mathcal{S})$, so we effectively only have to manage $|\pi_1(\mathcal{S})| + |\mathcal{S}^c|$ subroutines.

2.4 Conditional Correlated Equilibrium and ϵ-Dominated Actions

It is well-known that regret minimization in online learning is related to game-theoretic equilibria Nisan et al. [2007]. Specifically, when both players in a two-player zero-sum game follow external regret minimizing strategies, then the product of their individual empirical distributions converges to a Nash equilibrium. Moreover, if all players in a general K-player game follow swap regret minimizing strategies, then their empirical joint distribution converges to a correlated equilibrium Cesa-Bianchi and Lugosi [2006].

We will show in this section that when all players follow conditional swap regret minimization strategies, then the empirical joint distribution will converge to a new stricter type of correlated equilibrium.

Definition 2.1. Let $\Sigma_k = \{1, \ldots, N_k\}$, for $k \in \{1, \ldots, K\}$ and $G = (\mathcal{S} = \times_{k=1}^{K} \Sigma_k, \{l^{(k)} \colon \mathcal{S} \to [0,1]\}_{k=1}^{K})$ denote a K-player game. Let $s = (s_1, \ldots, s_K) \in \mathcal{S}$ denote the strategies of all players in one instance of the game, and let $s_{(-k)}$ denote the $(K-1)$-vector of strategies played by all players aside from player k. A joint distribution P on two rounds of this game is a **conditional correlated equilibrium** if for any player k, actions $j, j' \in \Sigma_k$, and map $\varphi_k \colon \Sigma_k^2 \to \Sigma_k$, we have

$$\sum_{(s,r) \in \mathcal{S}^2 \colon s_k = j, r_k = j'} P(s,r) \left(l^{(k)}(s_k, s_{(-k)}) - l^{(k)}(\varphi_k(s_k, r_k), s_{(-k)}) \right) \leq 0.$$

The standard interpretation of correlated equilibrium, which was first introduced by Aumann, is a scenario where an external authority assigns mixed strategies to each player in such a way that no player has an incentive to deviate from the recommendation, provided that no other player deviates from his Aumann [1974]. In the context of repeated games, a conditional correlated equilibrium is a situation where an external authority assigns mixed strategies to each player in such a way that no player has an incentive to deviate from the recommendation in the second round, even after factoring in information from the previous round of the game, provided that no other player deviates from his.

It is important to note that the concept of conditional correlated equilibrium presented here is different from the notions of extensive form correlated equilibrium and repeated game correlated equilibrium that have been studied in the game theory and economics literature Forges [1986], Lehrer [1992].

Notice that when the values taken for φ_k are independent of its second argument, we retrieve the familiar notion of correlated equilibrium.

Theorem 2.5. *Suppose that all players in a K-player repeated game follow bigram*

conditional swap regret minimizing strategies. Then, the joint empirical distribution of all players converges to a conditional correlated equilibrium.

Proof. Let $I^t \in \mathcal{S}$ be a random vector denoting the actions played by all K players in the game at round t. The empirical joint distribution of every two subsequent rounds of a K-player game played repeatedly for T total rounds has the form $\widehat{P}^T = \frac{1}{T}\sum_{t=1}^{T}\sum_{(s,r)\in\mathcal{S}^2}\delta_{\{I^t=s, I^{t-1}=r\}}$, where $I=(I_1,\ldots,I_K)$ and $I_k \sim \mathsf{p}^{(k)}$ denotes the action played by player k using the mixed strategy $\mathsf{p}^{(k)}$.

Then, the conditional swap regret of each player k, $\mathrm{reg}(k,T)$, can be bounded as follows since he is playing with a conditional swap regret minimizing strategy:

$$\begin{aligned}
&\mathrm{reg}(k,T)\\
&= \frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{s_k^t\sim\mathsf{p}_t(k)}\left[l^{(k)}(s_k^t, s_{(-k)}^t)\right] - \min_{\varphi}\frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{s_k^t\sim\mathsf{p}_t^{(k)}}\left[l^{(k)}(\varphi(s_k^t, I_k^{t-1}), s_{(-k)}^t)\right]\\
&\leq \mathcal{O}\left(N\sqrt{\frac{\log(N)}{T}}\right).
\end{aligned}$$

Define the instantaneous conditional swap regret vector as

$$\widehat{r}_{t,j_0,j_1}^{(k)} = \delta_{\{I_{(k)}^t=j_0, I_{(k)}^{t-1}=j_1\}}\left(l^{(k)}\big(I^t\big) - l^{(k)}\big(\varphi_k(j_0,j_1), I_{(-k)}^t\big)\right),$$

and the expected instantaneous conditional swap regret vector as

$$r_{t,j_0,j_1}^{(k)} = \mathbb{P}(s_k^t=j_0)\delta_{\{I_{(k)}^{t-1}=j_1\}}\left(l^{(k)}\big(j_0, I_{(-k)}^t\big) - l^{(k)}\big(\varphi_k(j_0,j_1), I_{(-k)}^t\big)\right).$$

Consider the filtration $\mathcal{G}_t = \{\text{information of opponents at time } t \text{ and of the player's actions up to time } t-1\}$. Then, we see that $\mathbb{E}\left[\widehat{r}_{t,j_0,j_1}^{(k)}|\mathcal{G}_t\right] = r_{t,j_0,j_1}^{(k)}$. Thus, $\{R_t = r_{t,j_0,j_1}^{(k)} - \widehat{r}_{t,j_0,j_1}^{(k)}\}_{t=1}^{\infty}$ is a sequence of bounded martingale differences, and by the

Hoeffding-Azuma inequality, we can write for any $\alpha > 0$, that $\mathbb{P}[|\sum_{t=1}^{T} R_t| > \alpha] \leq 2\exp(-C\alpha^2/T)$ for some constant $C > 0$.

Now define the sets $A_T := \left\{ \left| \frac{1}{T}\sum_{t=1}^{T} R_t \right| > \sqrt{\frac{C}{T}\log\left(\frac{2}{\delta_T}\right)} \right\}$. By our concentration bound, we have $P(A_T) \leq \delta_T$. Setting $\delta_T = \exp(-\sqrt{T})$ and applying the Borel-Cantelli lemma, we obtain that $\limsup_{T\to\infty} |\frac{1}{T}\sum_{t=1}^{T} R_t| = 0$ a.s..

Finally, since each player followed a conditional swap regret minimizing strategy, we can write $\limsup_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \widehat{r}^{(k)}_{t,j_0,j_1} \leq 0$. Now, if the empirical distribution did not converge to a conditional correlated equilibrium, then by Prokhorov's theorem, there exists a subsequence $\{\widehat{P}^{T_j}\}_j$ satisfying the conditional correlated equilibrium inequality but converging to some limit P^* that is not a conditional correlated equilibrium. This cannot be true because the inequality is closed under weak limits. □

Convergence to equilibria over the course of repeated game-playing also naturally implies the scarcity of "very suboptimal" strategies.

Definition 2.2. An action pair $(s_k, r_k) \in \Sigma_k^2$ played by player k is **conditionally ϵ-dominated** if there exists a map $\varphi_k \colon \Sigma_k^2 \to \Sigma_k$ such that

$$l^{(k)}(s_k, s_{(-k)}) - l^{(k)}(\varphi_k(s_k, r_k), s_{(-k)}) \geq \epsilon.$$

Theorem 2.6. *Suppose player k follows a conditional swap regret minimizing strategy that produces a regret R over T instances of the repeated game. Then, on average, an action pair of player k is conditionally ϵ-dominated at most $\frac{R}{\epsilon T}$ fraction of the time.*

The proof of this result is provided in Appendix B.4.

2.5 Bandit Scenario

As discussed earlier, the bandit scenario differs from the full-information scenario in that the player only receives information about the loss of his action $f_t(x_t)$ at each time and not the entire loss function f_t. One standard external regret minimizing algorithm is the Exp3 algorithm introduced by Auer et al. [2002], and it is the base learner off of which we will build a conditional swap regret minimizing algorithm.

To derive a sublinear conditional swap regret bound, we require an external regret bound on Exp3:

$$\sum_{t=1}^{T} \mathop{\mathbb{E}}_{p_t}[f_t(x_t)] - \min_{j \in \Sigma} \sum_{t=1}^{T} f_t(j) \leq 2\sqrt{L_{\min} N \log(N)},$$

which can be found in Theorem 3.1 of Bubeck et al. [2012]. Using this estimate, we can derive the following result.

Theorem 2.7. *There exists an algorithm $\mathcal{A}$ such that*

$$\mathrm{Reg}_{T,bandit}(\mathcal{C}_2, \mathcal{A}) \leq \mathcal{O}\big(\sqrt{N^3 \log(N) T}\big).$$

The proof is given in Appendix B.5 and is very similar to the proof for the full information setting.

It can also easily be extended in the analogous way to provide a regret bound for the k-gram regret in the bandit scenario.

Theorem 2.8. *There exists an algorithm $\mathcal{A}$ such that*

$$\mathrm{Reg}_{T,bandit}(\mathcal{C}_k, \mathcal{A}) \leq \mathcal{O}\big(\sqrt{N^{k+1} \log(N) T}\big).$$

See Appendix B.6 for an outline of the algorithm.

2.6 Summary of chapter

We analyzed the extent to which online learning scenarios are learnable. In contrast to some of the more recent work that has focused on increasing the power of the adversary (see e.g. Arora et al. [2012]), we increased the power of the competitor class instead by allowing history-dependent action swaps and thereby extending the notion of swap regret. We proved that this stronger class of competitors can still be beaten in the sense of sublinear regret as long as the memory of the competitor is bounded. We also provided a state-dependent bound that gives a more favorable guarantee when only some parts of the history are considered. In the bigram setting, we introduced the notion of conditional correlated equilibrium in the context of repeated K-player games, and showed how it can be seen as a generalization of the traditional correlated equilibrium. We proved that if all players follow bigram conditional swap regret minimizing strategies, then the empirical joint distribution converges to a conditional correlated equilibrium and that no player can play very suboptimal strategies too often. Finally, we showed that sublinear conditional swap regret can also be achieved in the partial information bandit setting.

Chapter 3

Adaptive and Optimistic Online Convex Optimization

In this chapter, we present a general framework for designing data-dependent online convex optimization algorithms, building upon and unifying recent techniques in adaptive regularization, optimistic gradient predictions, and problem-dependent randomization. We first present a series of new regret guarantees that hold at any time and under very minimal assumptions, and then show how different relaxations recover existing algorithms, both basic as well as more recent sophisticated ones. Finally, we show how combining adaptivity, optimism, and problem-dependent randomization can guide the design of algorithms that benefit from more favorable guarantees than recent state-of-the-art methods.

3.1 Preliminaries

In the standard scenario of online convex optimization [Zinkevich, 2003], at each round $t = 1, 2, \ldots$, the learner selects a point x_t out of a compact convex set $\mathcal{K} \subset \mathbb{R}^n$ and incurs loss $f_t(x_t)$, where f_t is a convex function defined over $\mathcal{K}$. The learner's objective is to find an algorithm $\mathcal{A}$ that minimizes the regret with respect to the best fixed point $x \in \mathcal{K}$:

$$\mathrm{Reg}_T(\mathcal{A}) = \max_{x \in \mathcal{K}} \mathrm{Reg}_T(\mathcal{A}, x), \quad \text{where } \mathrm{Reg}_T(\mathcal{A}, x) = \sum_{t=1}^{T} f_t(x_t) - f_t(x).$$

We will assume that the learner has access to the gradient or an element of the sub-gradient of the loss functions f_t, but that the loss functions f_t can be arbitrarily singular and flat, e.g. not necessarily strongly convex or smooth. This is the most general setup of convex optimization in the full information setting. It can be applied to standard convex optimization and online learning tasks as well as to many optimization problems in machine learning such as those of SVMs, logistic regression, and ridge regression. Favorable bounds in online convex optimization can also be translated into strong learning guarantees in the standard scenario of batch supervised learning using online-to-batch conversion guarantees [Littlestone, 1989, Cesa-Bianchi et al., 2004, Mohri et al., 2012].

In the scenario of online convex optimization just presented, minimax optimal rates can be achieved by standard algorithms such as online gradient descent [Zinkevich, 2003]. However, general minimax optimal rates may be too conservative. Recently, *adaptive regularization* methods have been introduced for standard descent methods to achieve tighter data-dependent regret bounds (see [Bartlett et al., 2007, Duchi et al., 2010, McMahan and Streeter, 2010, McMahan, 2014, Orabona

et al., 2015]). Specifically, in the AdaGrad algorithm of Duchi et al. [2010], the authors design a sequence of convex functions ψ_t such that the update $x_{t+1} = \operatorname{argmin}_{x \in \mathcal{K}} \eta g_t^\top x + B_{\psi_t}(x, x_t)$ achieves regret:

$$\mathrm{Reg}_T(\textsc{AdaGrad}, x) \leq \sqrt{2} \max_t \|x - x_t\|_\infty \sum_{i=1}^n \sqrt{\sum_{t=1}^T |g_{t,i}|^2},$$

where $g_t \in \partial f_t(x_t)$ is an element of the subgradient of f_t at x_t, $g_{1:T,i} = \sum_{t=1}^T g_{t,i}$, and B_{ψ_t} is the Bregman divergence defined using the convex function ψ_t. This upper bound on the regret has shown to be within a factor $\sqrt{2}$ of the a posteriori optimal regret with respect to a family of quadratic regularizers:

$$\mathrm{Reg}_T(\textsc{AdaGrad}, x) \leq 2 \max_t \|x - x_t\|_\infty \sqrt{n \inf_{s \succcurlyeq 0, 1^\top s \leq n} \sum_{t=1}^T \|g_t\|^2_{\mathrm{diag}(s)^{-1}}}.$$

The comparison family demonstrated here is the set of all non-negative diagonal matrices with bounded trace, which is both computationally efficient and the most commonly used benchmark in practice. Note, however, that this upper bound on the regret can still be very large, even if the functions f_t admit some favorable properties (e.g. $f_t = f$, linear). This is because the dependence is directly on the norm of g_ts.

An alternative line of research has been investigated by a series of recent publications that have analyzed online learning in "slowly-varying" scenarios [Hazan and Kale, 2009, Chiang et al., 2012, Rakhlin and Sridharan, 2013a,b, Chiang et al., 2013]. Rakhlin and Sridharan [2013b], in particular, introduce an algorithm called OptimisticFTRL that uses a sequence of *gradient predictions*. Specifically, if $\mathcal{R}$ is a self-concordant regularization function, $\|\cdot\|_{\nabla^2 \mathcal{R}(x_t)}$ is the semi-norm

induced by its Hessian at the point x_t,[1] and $\tilde{g}_{t+1} = \tilde{g}_{t+1}(g_1, \ldots, g_t, x_1, \ldots, x_t)$ is a "prediction" of a time $t+1$ subgradient g_{t+1} based on information up to time t, then OptimisticFTRL achieves regret bounds of the following form:

$$\mathrm{Reg}_T(\textsc{OptimisticFTRL}, x) \leq \sqrt{4 \max_{x \in \mathcal{K}} \mathcal{R}(x) \left[\sum_{t=1}^{T} \|g_t - \tilde{g}_t\|^2_{\nabla^2 \mathcal{R}(x_t),*} + 1 \right]}.$$

Here, $\|\cdot\|_{\nabla^2 \mathcal{R}(x_t),*}$ denotes the dual norm of $\|\cdot\|_{\nabla^2 \mathcal{R}(x_t)}$: for any x, $\|x\|_{\nabla^2 \mathcal{R}(x_t),*} = \sup_{\|y\|_{\nabla^2 \mathcal{R}(x_t)} \leq 1} x^\top y$. This guarantee can be very favorable in the *optimistic* case where $\tilde{g}_t \approx g_t$ for all t. Nevertheless, it admits the drawback that the chosen regularization admits no guarantee of "near-optimality" with respect to a family of regularizers as with the adaptive algorithm.

The discussion above naturally compels one to ask whether it is possible to combine the two data-dependent methods into a technique that captures the desirable attributes of both while discarding the deficiencies. For instance, one would aspire to attain regret guarantees of the form:

$$\max_t \|x - x_t\|_\infty \sqrt{n \inf_{s \succcurlyeq 0, 1^\top s \leq n} \sum_{t=1}^{T} \|g_t - \tilde{g}_t\|^2_{\mathrm{diag}(s)^{-1}}}. \tag{3.1}$$

This paper develops a framework for designing algorithms that can achieve such guarantees. These algorithms combine adaptive regularization and optimistic gradient prediction to address several of the issues that have been pointed out. Our framework builds upon and unifies recent techniques in adaptive regularization and optimistic gradient predictions, and as an example, attains the regret bound (3.1). In Section 3.2, we describe a series of *adaptive and optimistic* algorithms for

[1] The norm induced by a symmetric positive definite (SPD) matrix A is defined for any x by $\|x\|_A = \sqrt{x^\top A x}$.

which we prove strong regret guarantees, including a new *Adaptive and Optimistic Follow-the-Regularized-Leader* (AdaOptFTRL) algorithm (Section 3.2.1) and a more general version of this algorithm with composite terms (Section 3.2.3). These new regret guarantees hold at any time and under very minimal assumptions. We also show how different relaxations recover both basic existing algorithms as well as more recent sophisticated ones. In a specific application, we will also show how a certain choice of regularization functions will produce an optimistic regret bound that is also nearly a posteriori optimal, combining the two different desirable properties mentioned above. Lastly, in Section 3.3, we further analyze adaptivity and optimism in the stochastic optimization framework, constructing algorithms for random coordinate descent and stochastic empirical risk minimization that benefit from more favorable guarantees than recent state-of-the-art methods.

3.2 Adaptive and Optimistic Follow-the-Regularized-Leader algorithms

3.2.1 AdaOptFTRL algorithm

In view of the discussion in the previous section, we present an adaptive and optimistic version of the Follow-the-Regularized-Leader (FTRL) family of algorithms. In each round of standard FTRL, a point is chosen that is the minimizer of the average linearized loss incurred plus a regularization term. In our new version of FTRL, we will find a minimizer of not only the average loss incurred, but also a prediction of the next round's loss. In addition, we will define a dynamic time-varying sequence of regularization functions that can be used to optimize

Algorithm 15: AdaOptFTRL

Algorithm: AdaOptFTRL(r_0),
r_0 regularization function.
$\tilde{g}_1 \leftarrow 0$.
$x_1 \leftarrow \operatorname{argmin}_{x \in \mathcal{K}} r_0(x)$.
for $t \leftarrow 1, \dots, T$ **do**
  $g_t \leftarrow \textsc{ComputeSubgradient}(f_t, x_t)$.
  $r_t \leftarrow \textsc{DesignRegularizer}((g_s)_{s=1}^t, (x_s)_{s=1}^t)$.
  $\tilde{g}_{t+1} \leftarrow \textsc{PredictGradient}((g_s)_{s=1}^t, (x_s)_{s=1}^t)$.
  $x_{t+1} \leftarrow \operatorname*{argmin}_{x \in \mathcal{K}} g_{1:t}^\top x + \tilde{g}_{t+1}^\top x + r_{0:t}(x)$.

against this new loss term. Algorithm 15 shows the pseudocode of our Adaptive and Optimistic Follow-the-Regularized-Leader algorithm, AdaOptFTRL.

Note here that we have subsumed the step-size that is common to many incremental gradient methods into the regularization function. This is because we will later tune our regularization in a problem-dependent way to yield state-of-the-art guarantees. Combining the regularization with the step-size avoids the need to handle an additional parameter.

In the rest of the paper, we say that a regularization function $r_t : \mathbb{R}^n \to \mathbb{R}$ corresponding to an algorithm is *proximal* if $\operatorname{argmin}_{x \in \mathcal{K}} r_t(x) = x_t$. The following result provides a regret guarantee for AdaOptFTRL when using such regularizers.

Theorem 3.1 (AdaOptFTRL guarantee for proximal regularization)**.**
Let $\{r_t\}_{t=1}^T$ be a sequence of proximal non-negative functions, and let $\tilde{g}_t$ be the learner's estimate of g_t given the history of functions $(f_s)_{s=1}^{t-1}$ and points $(x_s)_{s=1}^{t-1}$. Assume further that the function $h_{0:t} \colon x \mapsto g_{1:t}^\top x + \tilde{g}_{t+1}^\top x + r_{0:t}(x)$ is 1-strongly convex with respect to some norm $\|\cdot\|_{(t)}$ (i.e. $r_{0:t}$ is 1-strongly convex with respect to

$\|\cdot\|_{(t)}$). *Then, the following regret bound holds for* AdaOptFTRL *(Algorithm 15):*

$$\mathrm{Reg}_T(\textsc{AdaOptFTRL}, x) = \sum_{t=1}^{T} f_t(x_t) - f_t(x) \le r_{0:T}(x) + \sum_{t=1}^{T} \|g_t - \tilde{g}_t\|_{(t),*}^2 .$$

Proof. Recall that $x_{t+1} = \operatorname{argmin}_{x\in\mathcal{K}} (g_{1:t} + \tilde{g}_{t+1})^\top x + r_{0:t}(x) = \operatorname{argmin}_{x\in\mathcal{K}} h_{0:t}(x)$, and let $y_t = \operatorname{argmin}_{x\in\mathcal{K}} x^\top g_{1:t} + r_{0:t}(x)$. Then, by convexity, the following inequality holds:

$$\begin{aligned}\sum_{t=1}^{T} f_t(x_t) - f_t(x) &\le \sum_{t=1}^{T} g_t^\top (x_t - x) \\ &= \sum_{t=1}^{T} (g_t - \tilde{g}_t)^\top (x_t - y_t) + \tilde{g}_t^\top (x_t - y_t) + g_t^\top (y_t - x).\end{aligned}$$

Now, we first prove by induction on T that for all $x \in \mathcal{K}$ the following inequality holds:

$$\sum_{t=1}^{T} \tilde{g}_t^\top (x_t - y_t) + g_t^\top y_t \le \sum_{t=1}^{T} g_t^\top x + r_{0:T}(x).$$

For $T = 1$, since $\tilde{g}_1 = 0$ and $r_1 \ge 0$, the inequality follows by the definition of y_1.

Now, suppose the inequality holds at iteration T. Then, we can write

$$
\begin{aligned}
&\sum_{t=1}^{T+1} \tilde{g}_t^\top (x_t - y_t) + g_t^\top y_t \\
&= \left[\sum_{t=1}^{T} \tilde{g}_t^\top (x_t - y_t) + g_t^\top y_t \right] + \tilde{g}_{T+1}^\top (x_{T+1} - y_{T+1}) + g_{T+1}^\top y_{T+1} \\
&\leq \left[\sum_{t=1}^{T} g_t^\top x_{T+1} + r_{0:T}(x_{T+1}) \right] + \tilde{g}_{T+1}^\top (x_{T+1} - y_{T+1}) + g_{T+1}^\top y_{T+1} \\
&\quad \text{(by the induction hypothesis for } x = x_{T+1}) \\
&\leq \left[(g_{1:T} + \tilde{g}_{T+1})^\top x_{T+1} + r_{0:T+1}(x_{T+1}) \right] + \tilde{g}_{T+1}^\top (-y_{T+1}) + g_{T+1}^\top y_{T+1} \\
&\quad \text{(since } r_t \geq 0, \, \forall t) \\
&\leq \left[(g_{1:T} + \tilde{g}_{T+1})^\top y_{T+1} + r_{0:T+1}(y_{T+1}) \right] + \tilde{g}_{T+1}^\top (-y_{T+1}) + g_{T+1}^\top y_{T+1} \\
&\quad \text{(by definition of } x_{T+1}) \\
&\leq g_{1:T+1}^\top y + r_{0:T+1}(y), \text{ for any } y \\
&\quad \text{(by definition of } y_{T+1}).
\end{aligned}
$$

Thus, we $\sum_{t=1}^{T} f_t(x_t) - f_t(x) \leq r_{0:T}(x) + \sum_{t=1}^{T} (g_t - \tilde{g}_t)^\top (x_t - y_t)$ and it suffices to bound $\sum_{t=1}^{T} (g_t - \tilde{g}_t)^\top (x_t - y_t)$. Notice that, by duality, one can immediately write $(g_t - \tilde{g}_t)^\top (x_t - y_t) \leq \|g_t - \tilde{g}_t\|_{(t),*} \|x_t - y_t\|_{(t)}$. To bound $\|x_t - y_t\|_{(t)}$ in terms of the gradient, recall first that since r_t is proximal and $x_t = \operatorname{argmin}_{x \in \mathcal{K}} h_{0:t-1}$,

$$
\begin{aligned}
x_t &= \operatorname*{argmin}_{x \in \mathcal{K}} h_{0:t-1}(x) + r_t(x), \\
y_t &= \operatorname*{argmin}_{x \in \mathcal{K}} h_{0:t-1}(x) + r_t(x) + (g_t - \tilde{g}_t)^\top x.
\end{aligned}
$$

The fact that $r_{0:t}(x)$ is 1-strongly convex with respect to the norm $\|\cdot\|_{(t)}$ implies that $h_{0:t-1} + r_t$ is as well. In particular, it is 1-strongly convex at the points x_t and y_t.

But this then implies that the conjugate function is 1-strongly smooth on the image of the gradient, including at $\nabla(h_{0:t-1}+r_t)(x_t) = 0$ and $\nabla(h_{0:t-1}+r_t)(y_t) = -(g_t-\tilde{g}_t)$ (see Lemma C.1 in the appendix or [Rockafellar, 1970] for a general reference), which means that

$$\|\nabla((h_{0:t-1}+r_t)^*)(-(g_t-\tilde{g}_t)) - \nabla((h_{0:t-1}+r_t)^*)(0)\|_{(t)} \leq \|g_t-\tilde{g}_t\|_{(t),*}.$$

Since $\nabla((h_{0:t-1}+r_t)^*)(-(g_t-\tilde{g}_t)) = y_t$ and $\nabla((h_{0:t-1}+r_t)^*)(0) = x_t$, we $\|x_t-y_t\|_{(t)} \leq \|g_t-\tilde{g}_t\|_{(t),*}$. □

The regret bound just presented can be much tighter than the adaptive methods of [Duchi et al., 2010, McMahan and Streeter, 2010], and others. For instance, one common choice of gradient prediction is $\tilde{g}_{t+1} = g_t$, so that for slowly varying gradients (e.g. functions that change "predictably"), $g_t-\tilde{g}_t \approx 0$, but $\|g_t\|_{(t)} = \|g\|_{(t)}$. Moreover, for reasonable gradient predictions, $\|\tilde{g}_{t+1}\|_{(t)} \approx \|g_t\|_{(t)}$ generally, so that in the worst case, the regret of AdaOptFTRL will be at most a factor of two more than standard methods. At the same time, the use of non-self-concordant regularization allows one to more explicitly control the induced norm in the regret bound and consequently learn faster per update than in the methods of [Rakhlin and Sridharan, 2013a]. Section 3.2.2.1 presents an upgraded version of online gradient descent as an example, where our choice of regularization allows our algorithm to *accelerate* as the gradient predictions become more accurate.

Note that the assumption of strong convexity of $h_{0:t}$ is not a significant constraint, as any standard quadratic or entropic regularizer used in the online mirror descent family of algorithms will satisfy this property.

Moreover, if the loss functions $\{f_t\}_{t=1}^T$ themselves are 1-strongly convex, then

one can set $r_{0:t} = 0$ and still get a favorable induced norm $\|\cdot\|^2_{(t),*} = \frac{1}{t}\|\cdot\|_2^2$. If the gradients and gradient predictions are uniformly bounded, this recovers the worst-case $\log(T)$ regret bounds. At the same time, Algorithm 15 would also still retain the potentially highly favorable data-dependent and optimistic regret bound.

Liang and Steinhardt (2014) [Steinhardt and Liang, 2014] also studied adaptivity and optimism in online learning in the context of mirror descent-type algorithms. If, in the proof above, we assume their condition:

$$r^*_{0:t+1}(-\eta g_{1:t}) \leq r^*_{0:t}(-\eta(g_{1:t} - \tilde{g}_t)) - \eta x_t^\top (g_t - \tilde{g}_t),$$

then we obtain the following regret bound:

$$\sum_{t=1}^{T} f_t(x_t) - \sum_{t=1}^{T} f_t(x) \leq \frac{r^*_1(0) + r_{0:T+1}(x)}{\eta}.$$

Our algorithm, however, is generally easier to use since it holds for any sequence of regularization functions and does not require checking for that condition.

In some cases, it may be preferable to use non-proximal adaptive regularization. It can be shown that non-adaptive non-proximal FTRL corresponds to the dual averaging algorithm. Thus, this scenario arises, for instance, when one wishes to use regularizers such as the negative entropy to derive algorithms from the Exponentiated Gradient (EG) family (see [Shalev-Shwartz, 2012] for background). We thus present the following theorem.

Theorem 3.2 (AdaOptFTRL guarantee for general regularization)**.** *Let $\{r_t\}_{t=1}^T$ be a sequence of non-negative functions, and let $\tilde{g}_t$ be the learner's estimate of g_t given the history of functions $f_1, \ldots, f_{t-1}$ and points $x_1, \ldots, x_{t-1}$. Assume further*

that the function $h_{0:t} \colon x \mapsto g_{1:t}^\top x + \tilde{g}_{t+1}^\top x + r_{0:t}(x)$ *is 1-strongly convex with respect to some norm* $\|\cdot\|_{(t)}$ *(i.e.* $r_{0:t}$ *is 1-strongly convex with respect to* $\|\cdot\|_{(t)}$*). Then, the following regret bound holds for* AO-FTRL *(Algorithm 15):*

$$\sum_{t=1}^{T} f_t(x_t) - f_t(x) \leq r_{0:T-1}(x) + \sum_{t=1}^{T} \|g_t - \tilde{g}_t\|_{(t-1),*}^2$$

The proof of this result is similar to the proof of Theorem 3.1 and given in Appendix C.1.2.

As in the case of proximal regularization, AdaOptFTRL applied to general regularizers still admits the same benefits over the standard adaptive algorithms. In particular, the above algorithm is an upgrade over any dual averaging algorithm.

Notice that in the regret bound for AdaOptFTRL with general regularization, the prediction error at time t is measured in the norm induced by the regularization chosen at time $t-1$. On the other hand, for proximal regularization the prediction error at time t is measured in terms of the norm induced by the regularization chosen at time t, after the gradient g_t is seen. On the other hand, the regret bound for general regularization only includes the sum of the first $T-1$ regularization functions while the regret bound for proximal regualrization includes the sum of all T regularization functions. This a subtle but important distinction between the two choices of regularization.

Corollary 3.1. *With the following suitable choices of the parameters in Theorem 3.3, the following regret bounds can be recovered:*

1. *Adaptive FTRL-Prox of [McMahan, 2014] (up to a constant factor of 2):* $\tilde{g} = 0$.

2. *Primal-Dual AdaGrad of [Duchi et al., 2010]:* $r_{0:t} = \psi_t$, $\tilde{g} = 0$.

3. *Optimistic FTRL of [Rakhlin and Sridharan, 2013a]:* $r_0 = \eta\mathcal{R}$ *where* $\eta > 0$ *and* $\mathcal{R}$ *a self-concordant function,* $r_t = \psi_t = 0, \forall t \geq 1$.

3.2.2 Applications

3.2.2.1 AdaOptOGD algorithm, Adaptive and Optimistic Online Gradient Descent

Assume that our convex action set $\mathcal{K}$ is contained in an n-dimensional rectangle: $\mathcal{K} \subset \times_{i=1}^{n}[-R_i, R_i]$. Denote $\Delta_{s,i} = \sqrt{\sum_{a=1}^{s}(g_{a,i} - \widetilde{g}_{a,i})^2}$, and set

$$r_{0:t}(x) = \sum_{i=1}^{n}\sum_{s=1}^{t}\frac{\Delta_{s,i}-\Delta_{s-1,i}}{2R_i}(x_i - x_{s,i})^2.$$

Moreover, consider the martingale-type gradient prediction: $\tilde{g}_{t+1} = g_t$. We call AdaOptFTRL with this choice of regularization and gradient prediction AdaOptOGD for Adaptive and Optimistic Online Gradient Descent.

Corollary 3.2 (AdaOptOGD guarantee)**.** AdaOptOGD *achieves the following regret bound:*

$$Reg_T(\text{AdaOptOGD}, x) \leq 4\sum_{i=1}^{n} R_i\sqrt{\sum_{t=1}^{T}(g_{t,i} - g_{t-1,i})^2}.$$

Moreover, this regret bound is nearly a posteriori optimal over a family of quadratic regularizers :

$$\sum_{i=1}^{n} R_i\sqrt{\sum_{t=1}^{T}(g_{t,i} - g_{t-1,i})^2} = \max_i R_i\sqrt{n\inf_{s\succcurlyeq 0,\langle s,1\rangle\leq n}\sum_{t=1}^{T}\|g_t - g_{t-1}\|^2_{\text{diag}(s)^{-1}}}.$$

Proof. $r_{0:t}$ is 1-strongly convex with respect to the norm:

$$\|x\|_{(t)}^2 = \sum_{i=1}^{n} \frac{\sqrt{\sum_{a=1}^{t}(g_{a,i} - \tilde{g}_{a,i})^2}}{R_i} x_i^2,$$

which has corresponding dual norm:

$$\|x\|_{(t),*}^2 = \sum_{i=1}^{n} \frac{R_i}{\sqrt{\sum_{a=1}^{t}(g_{a,i} - \tilde{g}_{a,i})^2}} x_i^2.$$

By the choice of this regularization, the prediction $\tilde{g}_t = g_{t-1}$, and Theorem 3.3, the following holds:

$$\begin{aligned}
\mathrm{Reg}_T(\mathcal{A}, x) &\le \sum_{i=1}^{n}\sum_{s=1}^{T} \frac{\sqrt{\sum_{a=1}^{s}(g_{a,i} - \tilde{g}_{a,i})^2} - \sqrt{\sum_{a=1}^{s-1}(g_{a,i} - \tilde{g}_{a,i})^2}}{2R_i}(x_i - x_{s,i})^2 \\
&\quad + \sum_{t=1}^{T} \|g_t - g_{t-1}\|_{(t),*}^2 \\
&\le \sum_{i=1}^{n} 2R_i \sqrt{\sum_{t=1}^{T}(g_{t,i} - g_{t-1,i})^2} + \sum_{i=1}^{n}\sum_{t=1}^{T} \frac{R_i(g_{t,i} - g_{t-1,i})^2}{\sqrt{\sum_{a=1}^{t}(g_{a,i} - g_{a-1,i})^2}} \\
&\le \sum_{i=1}^{n} 2R_i \sqrt{\sum_{t=1}^{T}(g_{t,i} - g_{t-1,i})^2} + \sum_{i=1}^{n} 2R_i \sqrt{\sum_{t=1}^{T}(g_{t,i} - g_{t-1,i})^2} \\
&\quad \text{by Lemma C.2.}
\end{aligned}$$

The last statement follows from the fact that

$$\inf_{s \succcurlyeq 0, \langle s,1,\rangle \le n} \sum_{t=1}^{T}\sum_{i=1}^{n} \frac{g_{t,i}^2}{s_i} = \frac{1}{n}\left(\sum_{i=1}^{n} \|g_{1:T}, i\|_2\right)^2,$$

since the infimum on the left hand side is attained when $s_i \propto \|g_{1:T,i}\|_2$. $\square$

AdaOptOGD combines the adaptive properties of AdaGrad with the improvement under slowly-varying gradients that is enjoyed by OptimisticFTRL.

Notice that the regularization function is minimized when the gradient predictions become more accurate. Thus, if we interpret our regularization as an implicit learning rate, our algorithm uses a larger learning rate and *accelerates* as our gradient predictions become more accurate. This is in stark contrast to other adaptive regularization methods, such as AdaGrad, where learning rates are inversely proportional to simply the norm of the gradient.

Moreover, since the regularization function decomposes over the coordinates, this acceleration can occur on a per-coordinate basis. If our gradient predictions are more accurate in some coordinates than others, then our algorithm will be able to adapt accordingly. Under the simple martingale prediction scheme, this means that our algorithm will be able to adapt well when only certain coordinates of the gradient are slowly-varying, even if the entire gradient is not.

As an illustrative example, suppose we are trying to predict the gradient and did so to varying degrees of accuracy per coordinate. Specifically, let $A_k = |\{i \in [1, n]|\frac{1}{2^{k+1}} \leq \sum_{s=1}^{T}(g_{s,i} - \tilde{g}_{s,i})^2 < \frac{1}{2^k}\}|$, and assume that A_k is roughly constant across k such that $\forall k, j,\ A_k \leq CA_j$ for some mild universal constant C. Then the bound in Corollary 3.2 can be upper bounded by $\sum_{k=0}^{\infty} A_k\sqrt{\frac{1}{2^k}} = A_1 C\frac{\sqrt{2}}{\sqrt{2}-1}$, while standard learning algorithms with predictable sequences will incur a regret of $\sqrt{n}\sqrt{\sum_{k=0}^{\infty} A_k\frac{1}{2^k}} = A_1\sqrt{n}\sqrt{2}$, so that we improve by a $\sqrt{n}$ factor.

Moreover, since it is unlikely that the gradient predictions will ever be uniformly accurate across coordinates, some degree of the above improvement should take place for any model and problem.

Furthermore, notice that if we are in the sparse case in which AdaGrad performs well (see Duchi et al. [2010], Section 1.3), the martingale-type predictor $\tilde{g}_t = g_{t-1}$ will automatically tune the regularization of Algorithm C.1.5 into an AdaGrad-type one.

In terms of computation, the update in AdaOptOGD can be executed in time linear in the dimension (the same as for standard online gradient descent). Moreover, since the gradient prediction is simply the last gradient received, the algorithm also does not require much more storage than the standard gradient descent algorithm. However, as we mentioned in the general case, the regret bound here can be significantly more favorable than the standard $\mathcal{O}\left(\sqrt{T \sup_{t\in[1,T} \|g_t\|_2^2 \sum_{i=1}^n R_i^2]}\right)$ bound of online gradient descent, or even its adaptive variants.

3.2.3 CoAdaOptFTRL algorithm

In some cases, we may wish to impose some regularization on our original optimization problem to ensure properties such as generalization (e.g. l_2-norm in SVM) or sparsity (e.g. l^1-norm in Lasso). This *composite term* can be treated directly by modifying the regularization in our FTRL update. However, if we wish for the regularization penalty to appear in the regret expression but do not wish to linearize it (which could mitigate effects such as sparsity), then some extra care needs to be taken.

We incorporate this composite term into AdaOptFTRL and present CoAdaOptFTRL. We provide accompanying regret bounds for this algorithm with both proximal and general regularization functions. The latter is presented in Appendix C.1 (as well as all accompanying proofs). For each choice of regularization, we give a pair of regret bounds, depending on whether the learner considers the

Algorithm 16: CoAdaOptFTRL.

Algorithm: CoAdaOptFTRL(r_0, $\{\psi_t\}_{t=1}^T$),
r_0 regularization function, $\{\psi_t\}_{t=1}^T$ composite functions.
$\tilde{g}_1 \leftarrow 0$.
$x_1 \leftarrow \operatorname{argmin}_{x\in\mathcal{K}} r_0(x)$.
for $t = 1, \ldots, T$ **do**
  $g_t \leftarrow$ ComputeSubgradient(f_t, x_t).
  $r_t \leftarrow$ DesignRegularizer$((g_s)_{s=1}^t, (x_s)_{s=1}^t)$.
  $\tilde{g}_{t_1} \leftarrow$ PredictGradient$((g_s)_{s=1}^t, (x_s)_{s=1}^t)$.
  $x_{t+1} \leftarrow \operatorname*{argmin}_{x\in\mathcal{K}} g_{1:t}^\top x + \tilde{g}_{t+1}^\top x + r_{0:t}(x) + \psi_{1:t+1}(x)$.

composite term as an additional part of the loss.

Theorem 3.3 (CoAdaOptFTRL guarantee for proximal regularization)**.** *Let* $\{r_t\}_{t=1}^T$ *be a sequence of proximal non-negative functions, such that* $\operatorname{argmin}_{x\in\mathcal{K}} r_t(x) = x_t$*, and let* $\tilde{g}_t$ *be the learner's estimate of* g_t *given the history of functions* $(f_s)_{s=1}^{t-1}$ *and points* $(x_s)_{s=1}^{t-1}$*. Let* $\{\psi_t\}_{t=1}^T$ *be a sequence of non-negative convex functions, such that* $\psi_1(x_1) = 0$*. Assume further that the function* $h_{0:t} : x \mapsto g_{1:t}^\top x + \tilde{g}_{t+1}^\top x + r_{0:t}(x) + \psi_{1:t+1}(x)$ *is 1-strongly convex with respect to some norm* $\|\cdot\|_{(t)}$*. Then the following regret bounds hold for* CoAdaOptFTRL*:*

$$\sum_{t=1}^T f_t(x_t) - f_t(x) \le \psi_{1:T-1}(x) + r_{0:T-1}(x) + \sum_{t=1}^T \|g_t - \tilde{g}_t\|_{(t-1),*}^2$$
$$\sum_{t=1}^T [f_t(x_t) + \psi_t(x_t)] - [f_t(x) + \psi_t(x)] \le r_{0:T}(x) + \sum_{t=1}^T \|g_t - \tilde{g}_t\|_{(t),*}^2 .$$

Notice that if we don't consider the composite term as part of our loss, then our regret bound resembles the form of that of AdaOptFTRL with general regularization. This is in spite of the fact that we are using proximal adaptive regularization. On the other hand, if the composite term is part of our loss, then our regret bound resembles the one obtained by AdaOptFTRL with proximal

regularization.

3.3 Adaptive and Optimistic Follow-the-Regularized-Leader algorithms with stochastic subgradients

3.3.1 CoAdaOptFTRL algorithm with stochastic subgradients

We now generalize our scenario to that of stochastic online convex optimization, where, instead of exact subgradient elements g_t, we receive only estimates. Specifically, we assume access to a sequence of vectors of the form $\widehat{g}_t$, where $\mathbb{E}[\widehat{g}_t|g_1,\dots,g_{t-1},x_1,\dots,x_t] = g_t$. This extension is in fact well-documented in the literature (see [Shalev-Shwartz, 2012] for a reference), and the extension of our adaptive and optimistic variant follows accordingly. For completeness, we provide a proof of the following theorem as well as its non-proximal analogue in Appendix C.2.

Theorem 3.4 (CoAdaOptFTRL stochastic proximal guarantee)**.** *Let* $\{r_t\}_{t=1}^T$ *be a sequence of proximal non-negative functions, such that* $\operatorname{argmin}_{x\in\mathcal{K}} r_t(x) = x_t$*, and let* $\tilde{g}_t$ *be the learner's estimate of* $\widehat{g}_t$ *given the history of noisy gradients* $(\widehat{g}_s)_{s=1}^{t-1}$ *and points* $(x_s)_{s=1}^{t-1}$*. Let* $\{\psi_t\}_{t=1}^T$ *be a sequence of non-negative convex functions, such that* $\psi_1(x_1) = 0$*. Assume further that the function* $h_{0:t}(x) = \widehat{g}_{1:t}^\top x + \tilde{g}_{t+1}^\top x + r_{0:t}(x) + \psi_{1:t+1}(x)$ *is 1-strongly convex with respect to some norm* $\|\cdot\|_{(t)}$*. Then, the update* $x_{t+1} = \operatorname{argmin}_{x\in\mathcal{K}} h_{0:t}(x)$ *of* CoAdaOptFTRL *with stochastic subgradients yields*

the following regret bounds:

$$\mathbb{E}\left[\sum_{t=1}^{T} f_t(x_t) - f_t(x)\right] \leq \mathbb{E}\left[\psi_{1:T-1}(x) + r_{0:T-1}(x) + \sum_{t=1}^{T} \|\widehat{g}_t - \tilde{g}_t\|^2_{(t-1),*}\right]$$
$$\mathbb{E}\left[\sum_{t=1}^{T} f_t(x_t) + \psi_t(x_t) - f_t(x) - \psi_t(x)\right] \leq \mathbb{E}\left[r_{0:T}(x) + \sum_{t=1}^{T} \|\widehat{g}_t - \tilde{g}_t\|^2_{(t),*}\right].$$

The algorithm above enjoys the same advantages over its non-adaptive or non-optimistic predecessors. Moreover, the choice of the adaptive regularizers $\{r_t\}_{t=1}^T$ and gradient predictions $\{\tilde{g}\}_{t=1}^T$ now also depend on the randomness of the gradients received. While masked in the above regret bounds, this interplay will come up explicitly in the following two examples, where we, as the learner, impose randomness into the problem.

3.3.2 Applications

3.3.2.1 Randomized Coordinate Descent with Adaptive Probabilities

Randomized coordinate descent is a method that is often used for very large-scale problems where it is impossible to compute and/or store entire gradients at each step. It is also effective for directly enforcing sparsity in a solution since the support of the final point x_t cannot be larger than the number of updates introduced.

The standard randomized coordinate descent update is to choose a coordinate uniformly at random (see e.g. [Shalev-Shwartz and Tewari, 2011]). Nesterov (2012) [Nesterov, 2012] analyzed random coordinate descent in the context of loss functions with higher regularity and showed that one can attain better bounds by using non-uniform probabilities.

In the general framework, we specify, at each round t, a distribution p_t over the n coordinates and pick a coordinate $i_t \in \{1, \ldots, n\}$ randomly according to this distribution. We then construct an unbiased estimate of an element of the subgradient: $\widehat{g}_t = \frac{(g_t^\top e_{i_t}) e_{i_t}}{p_{t,i_t}}$. This *importance weighting* technique is common in the online learning literature, particularly in the context of the multi-armed bandit problem (see e.g. [Cesa-Bianchi and Lugosi, 2006] for more information). Moreover, given a gradient prediction $\tilde{g}_t$, we also modify this prediction using the same distribution and scheme: $\widehat{\tilde{g}}_t = \frac{(\tilde{g}_t^\top e_{i_t}) e_{i_t}}{p_{t,i_t}}$.

We call AdaOptOGD with these stochastic subgradients and subgradient predictions AdaOptRCD. By applying Theorem 3.4 to these stochastic gradients, we can derive the following regret guarantee for AdaOptRCD.

Theorem 3.5 (AdaOptRCD guarantee)**.** *Assume $\mathcal{K} \subset \times_{i=1}^n [-R_i, R_i]$. Let i_t be a random variable sampled according to the distribution p_t, and let*

$$\widehat{g}_t = \frac{(g_t^\top e_{i_t}) e_{i_t}}{p_{t,i_t}}, \quad \widehat{\tilde{g}}_t = \frac{(\tilde{g}_t^\top e_{i_t}) e_{i_t}}{p_{t,i_t}},$$

be the estimated gradient and estimated gradient prediction. Denote

$$\Delta_{s,i} = \sqrt{\sum_{a=1}^{s} (\widehat{g}_{a,i} - \widehat{\tilde{g}}_{a,i})^2},$$

and let

$$r_{0:t} = \sum_{i=1}^{n} \sum_{s=1}^{t} \frac{\Delta_{s,i} - \Delta_{s-1,i}}{2R_i} (x_i - x_{s,i})^2$$

be the adaptive regularization. Then, the regret of the algorithm can be bounded by:

$$\mathbb{E}\left[\sum_{t=1}^{T} f_t(x_t) + \psi_t(x_t) - f_t(x) - \psi_t(x)\right] \leq 4\sum_{i=1}^{n} R_i \sqrt{\sum_{t=1}^{T} \mathbb{E}\left[\frac{(g_{t,i} - \tilde{g}_{t,i})^2}{p_{t,i}}\right]}.$$

Proof. We can first compute

$$\mathbb{E}\left[\widehat{g}_t\right] = \mathbb{E}\left[\frac{(g_t^\top e_{i_t}) e_{i_t}}{p_{t,i_t}}\right] = \sum_{i=1}^{n} \frac{(g_t^\top e_i) e_i}{p_{t,i}} p_{t,i} = g_t,$$

and similarly for the gradient prediction $\tilde{g}_t$.

Now, as in Corollary 3.2, the choice of regularization ensures us a regret bound of the form:

$$\mathbb{E}\left[\sum_{t=1}^{T} f_t(x_t) + \psi(x_t) - f_t(x) - \psi(x)\right] \leq 4\sum_{i=1}^{n} R_i \, \mathbb{E}\left[\sqrt{\sum_{t=1}^{T} (\widehat{g}_{t,i} - \tilde{g}_{t,i})^2}\right].$$

Moreover, by Jensen's inequality, it follows that:

$$\mathbb{E}\left[\sqrt{\sum_{t=1}^{T} (\widehat{g}_{t,i} - \tilde{g}_{t,i})^2}\right] \leq \sqrt{\mathbb{E}\left[\sum_{t=1}^{T} \mathop{\mathbb{E}}_{i_t}[(\widehat{g}_{t,i} - \tilde{g}_{t,i})^2]\right]} = \sqrt{\sum_{t=1}^{T} \mathbb{E}\left[\frac{(g_{t,i} - \tilde{g}_{t,i})^2}{p_{t,i}}\right]}.$$

□

In general, we do not have access to an element of the subgradient g_t before we sample according to p_t. However, if we assume that we have some per-coordinate upper bound on an element of the subgradient uniform in time, i.e. $|g_{t,j}| \leq L_j$ $\forall t \in \{1, \ldots, T\}, j \in \{1, \ldots, n\}$, then we can use the fact that $|g_{t,j} - \tilde{g}_{t,j}| \leq \max\{L_j - \tilde{g}_{t,j}, \tilde{g}_{t,j}\}$ to motivate setting $\tilde{g}_{t,j} := \frac{L_j}{2}$ and $p_{t,j} = \frac{(R_j L_j)^{2/3}}{\sum_{k=1}^{n} (R_k L_k)^{2/3}}$ (by

computing the optimal distribution). This yields the following regret bound.

Corollary 3.3 (AdaOptRCD with Lipschitz loss functions guarantee)**.** *Assume that, at any time t, the following per-coordinate Lipschitz bounds hold on the loss function: $|g_{t,i}| \leq L_i, \quad \forall i \in \{1, \ldots, n\}$. Set $p_{t,i} = \frac{(R_i L_i)^{2/3}}{\sum_{j=1}^{n} (R_j L_j)^{2/3}}$ as the probability distribution at time t, and set $\tilde{g}_{t,i} = \frac{L_i}{2}$. Then, the regret of the algorithm can be bounded as follows:*

$$\mathbb{E}\left[\sum_{t=1}^{T} f_t(x_t) + \psi_t(x_t) - f_t(x) - \psi_t(x)\right] \leq 2\sqrt{T}\left(\sum_{i=1}^{n} (R_i L_i)^{2/3}\right)^{3/2}.$$

An application of Hölder's inequality will reveal that this bound is strictly smaller than the $2RL\sqrt{nT}$ bound one would obtain from randomized coordinate descent using the uniform distribution. As a motivating example, if the problem were poorly conditioned and there was only a single dominant coordinate i^*, then the bound of Corollary 3.3 would be $2\sqrt{T} R_{i^*} L_{i^*}$ while the standard bound would have an extra $\sqrt{n}$ factor.

Moreover, the algorithm above still entertains the intermediate data-dependent bound of Theorem 3.5.

Notice the similarity between the sampling distribution generated here with the one suggested by [Nesterov, 2012]. However, Nesterov assumed higher regularity in his algorithm (i.e. $f_t \in C^{1,1}$) and generated his probabilities using that assumption. In our setting, we only need $f_t \in C^{0,1}$. It should be noted that [Afkanpour et al., 2013] also proposed an importance-sampling based approach to random coordinate descent for the specific setting of multiple kernel learning. In their setting, they propose updating the sampling distribution at each point in time instead of using uniform-in-time Lipschitz constants, which comes with a natural computational

tradeoff. Moreover, the introduction of adaptive per-coordinate learning rates in our algorithm allows for tighter regret bounds in terms of the Lipschitz constants.

An intermediate point between using full gradient updates and sampling individual coordinates is to sample a mini-batch of coordinates. We can extend AdaOptRCD to this setting, which would have the following guarantee:

Corollary 3.4 (AdaOptRCD with Lipschitz losses and mini-batch sampling)**.** *Assume that* $\mathcal{K} \subset \times_{i=1}^{n}[-R_i, R_i]$. *Let*

$$\cup_{j=1}^{k}\{\Pi_j\} = \{1, \ldots, n\}$$

be a partition of the coordinates, and let $e_{\Pi_j} = \sum_{i \in \Pi_j} e_i$. *Assume we had the following Lipschitz condition on the partition:* $\|g_t^\top e_{\Pi_j}\| \leq L_j \ \forall j \in \{1, \ldots, k\}$.

Define $S_j = \sum_{i \in \Pi_j} R_i$. *Set* $p_{t,i} = \frac{(S_i L_i)^{2/3}}{\sum_{j=1}^{k}(S_j L_j)^{2/3}}$ *as the probability distribution at time* t, *and set* $\tilde{g}_{t,i} = \frac{L_i}{2}$.

Then the regret of the resulting algorithm is bounded by:

$$\mathbb{E}\left[\sum_{t=1}^{T} f_t(x_t) + \psi_t(x_t) - f_t(x) - \psi_t(x)\right] \leq 2\sqrt{T}\left(\sum_{i=1}^{k}(S_i L_i)^{2/3}\right)^{3/2}.$$

While the expression is similar to the non-mini-batch version, the L_i and R_i terms now have different meaning. Specifically, L_i is a bound on the 2-norm of the components of the gradient in each batch, and R_i is the 1-norm of the corresponding sides of the hypercube.

3.3.2.2 Stochastic Optimization for Regularized Empirical Risk Minimization

Many learning algorithms can be viewed as instances of regularized empirical risk minimization (e.g. SVM, Logistic Regression, Lasso), where the goal is to minimize an objective function of the following form:

$$H(x) = \sum_{j=1}^{m} f_j(x) + \lambda\psi(x).$$

If we denote the first term by $F(x) = \sum_{j=1}^{m} f_j(x)$, then we can view this objective in the CoAdaOptFTRL framework with stochastic subgradients, where $f_t = F$ and $\psi_t = \lambda\psi$. In the same spirit as for non-uniform random coordinate descent, we can estimate the gradient of H at x_t by sampling according to some distribution p_t and using importance weighting to generate an unbiased estimate. Specifically, if $g_t \in \partial F(x_t)$ and $g_t^j \in \partial f_j(x_t)$, then $g_t = \sum_{j=1}^{m} g_t^j \approx \frac{g_t^{j_t}}{p_{t,j_t}}$.

This motivates the design of an algorithm similar to the one derived for randomized coordinate descent. Here we elect to use as gradient prediction the last gradient of the current function being sampled f_j. However, we may run into the problem of never having seen a function before. A logical modification is to separate the optimization into K epochs and do a full batch update over all functions f_j at the start of each epoch. This is similar to the technique used in the Stochastic Variance Reduced Gradient (SVRG) algorithm of Johnson and Zhang [2013]. However, we do not assume extra function regularity as they do in their paper, so the bounds are not comparable. The algorithm is presented in Algorithm 17 and comes with the following guarantee:

Corollary 3.5 (AdaOptRegERM guarantee)**.** *Assume* $\mathcal{K} \subset \times_{i=1}^{n}[-R_i, R_i]$. *De-*

Algorithm 17: AdaOptRegERM

Algorithm: AdaOptRegERM(λ, r_0, ψ, $(p_t)_{t=1}^T$, K, x_1),
$\lambda > 0$ scaling constant, r_0 regularization function, ψ composite term, $(p_t)_{t=1}^T$ distributions on $\{1, \dots, m\}$, K number of epochs, x_1 initial point.

$j_1 \leftarrow \text{Sample}(p_1)$.
$t \leftarrow 1$.
for $s \leftarrow 1, \dots, k$ **do**
  for $j \leftarrow 1 \rightarrow m$ **do**
    $\bar{g}_s^j \leftarrow \nabla f_j(x_1)$.
  for $a \leftarrow 1, \dots, \frac{T}{k}$ **do**
    if $T \mod k = 0$ **then**
      for $j \leftarrow 1 \rightarrow m$ **do**
        $g^j \leftarrow \nabla f_j(x_t)$.
    $\widehat{g}_t \leftarrow \frac{g_t^{j_t}}{p_{t,j_t}}$.
    $r_t \leftarrow \text{DesignRegularizer}((\widehat{g}_s)_{s=1}^t, (x_s)_{s=1}^t)$.
    $j_{t+1} \leftarrow \text{Sample}(p_{t+1})$.
    $\tilde{g}_{t+1} \leftarrow \frac{\bar{g}_s^{j_t}}{p_{t,j_t}}$.
    $x_{t+1} \leftarrow \text{argmin}_{x \in \mathcal{K}} \widehat{g}_{1:t}^\top x + \tilde{g}_{t+1}^\top x + r_{0:t}(x) + (t+1)\lambda\psi(x)$.
    $t \leftarrow t + 1$.

note $\Delta_{s,i} = \sqrt{\sum_{a=1}^s (\widehat{g}_{a,i} - \tilde{g}_{a,i})^2}$, *and let* $r_{0:t} = \sum_{i=1}^n \sum_{s=1}^t \frac{\Delta_{s,i} - \Delta_{s-1,i}}{2R_i}(x_i - x_{s,i})^2$ *be the adaptive regularization.*

Then the regret of AdaOptRegERM *is bounded by:*

$$\mathbb{E}\left[\sum_{t=1}^T f_t(x_t) + \lambda\psi(x_t) - f_t(x) - \lambda\psi(x)\right] \leq \sum_{i=1}^n 4R_i \sqrt{\sum_{s=1}^K \sum_{t=\frac{(s-1)T}{K}+1}^{\frac{sT}{K}} \sum_{j=1}^m \frac{\left|g_{t,i}^j - \bar{g}_{s,i}^j\right|^2}{p_{t,j}}}.$$

Moreover, if $\|\nabla f_j\|_\infty \leq L_j \ \forall j$, *then setting* $p_{t,j} = \frac{L_i}{\sum_{j=1}^m L_j}$ *yields a worst-case bound of:* $8\sum_{i=1}^n R_i \sqrt{T\left(\sum_{j=1}^m L_j\right)^2}$.

We also include a mini-batch version of this algorithm in Appendix C.3, which

can be useful due to the variance reduction of the gradient prediction.

3.4 Summary of chapter

We presented a general framework for developing efficient adaptive and optimistic algorithms for online convex optimization. Building upon recent advances in adaptive regularization and predictable online learning, we improved upon each method. We demonstrated the power of this approach by deriving algorithms with better guarantees than those commonly used in practice. In addition, we also extended adaptive and optimistic online learning to the randomized setting. Here, we highlighted an additional source of problem-dependent adaptivity (that of prescribing the sampling distribution), and we showed how one can perform better than traditional naive uniform sampling.

Chapter 4

Structural Ensemble Methods for Online Learning

In this chapter, we study the problem of learning ensembles in the online setting, where the hypotheses are selected out of a base family that may be a union of possibly very complex sub-families. We prove new theoretical guarantees for the online learning of such ensembles in terms of the sequential Rademacher complexities of these sub-families. We also describe an algorithm that benefits from such guarantees. We further extend our framework by proving new structural estimation error guarantees for ensembles in the batch setting through a new data-dependent online-to-batch conversion technique, thereby also devising an effective algorithm for the batch setting which does not require the estimation of the Rademacher complexities of base sub-families.

Ensemble methods are powerful techniques in machine learning for combining several predictors to define a more accurate one. They include notable methods such as bagging and boosting [Breiman, 1996, Freund and Schapire, 1997], and

they have been successfully applied to a variety of scenarios including classification and regression.

Standard ensemble methods such as AdaBoost and Random Forests select base predictors from some hypothesis set $\mathcal{H}$, which may be the family of boosting stumps or that of decision trees with some limited depth. More complex base hypothesis sets may be needed to tackle some difficult modern tasks. At the same time, learning bounds for standard ensemble methods suggest a risk of overfitting when using very rich hypothesis sets, which has been further observed empirically [Dietterich, 2000, Rätsch et al., 2001].

Recent work in the batch setting has shown, however, that learning with such complex base hypothesis sets is possible using the *structure* of $\mathcal{H}$, that is its decomposition into subsets $\mathcal{H}_k$, $k = 1, \ldots, p$, of varying complexity. In particular, in [Cortes et al., 2014], the authors introduced a new ensemble algorithm, DeepBoost, which provably benefited from finer learning guarantees when using rich families as base classifier sets. In DeepBoost, the decisions in each iteration of which classifier to add to the ensemble and which weight to assign to that classifier depend on the complexity of the sub-family $\mathcal{H}_k$ to which the classifier belongs. This can be viewed as integrating the principle of structural risk minimization to each iteration of boosting.

This paper extends the *structural learning* idea of incorporating model selection in ensemble methods to the online learning setting. Specifically, we address the question: can one design ensemble algorithms for the online setting that admit strong guarantees even when using a complex $\mathcal{H}$? In Section 4.2, we first present a theoretical result guaranteeing the existence of a randomized algorithm that can compete against the best ensemble in $\mathcal{H}$ efficiently when this ensemble does not

rely *too heavily* on complex base hypotheses. Motivated by this theory, we then design an online algorithm that benefits from such guarantees for a wide family of hypotheses sets (Section 4.3). Finally, in Section 4.4, we further extend our framework by proving new structural estimation error guarantees for ensembles in the batch setting through a new data-dependent online-to-batch conversion technique. This also provides an effective algorithm for the batch setting which does not require the estimation of the Rademacher complexities of base hypothesis sets $\mathcal{H}_k$.

4.1 Preliminaries

Let $\mathcal{X}$ denote the input space and $\mathcal{Y}$ the output space. Let $L_t \colon \mathcal{Y} \to \mathbb{R}_+$ be a loss function. The online learning framework that we study is a sequential prediction setting that can be described as follows. At each time $t \in [1, T]$, the learner (or algorithm $\mathcal{A}$) receives an input instance x_t which he uses to select a hypothesis $h_t \in \mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ and make a prediction $h_t(x_t)$. The learner then incurs the loss $L_t(h_t(x_t))$ based on the loss function L_t chosen by an adversary. The objective of the learner is to minimize his regret over T rounds, that is the difference of his cumulative loss $\sum_{t=1}^{T} L_t(h_t(x_t))$ and that of the best function in some benchmark hypothesis set $\mathcal{F} \subset \mathcal{Y}^{\mathcal{X}}$:

$$\mathrm{Reg}_T(\mathcal{A}) = \sum_{t=1}^{T} L_t(h_t(x_t)) - \min_{h \in \mathcal{F}} \sum_{t=1}^{T} L_t(h(x_t)).$$

In what follows, $\mathcal{F}$ will be assumed to coincide with $\mathcal{H}$, unless explicitly stated otherwise. The learner's algorithm may be randomized, in which case, at each round t, the learner draws hypothesis h_t from the distribution p_t he has defined

at that round. The regret is then the difference between the expected cumulative loss and the expected cumulative loss of the best-in-class hypothesis: $\mathrm{Reg}_T(\mathcal{A}) = \sum_{t=1}^{T} \mathbb{E}[L_t(h_t(x_t))] - \min_{h\in\mathcal{H}} \sum_{t=1}^{T} \mathbb{E}[L_t(h(x_t))]$.

Clearly, the difficulty of the learner's regret minimization task depends on the richness of the competitor class $\mathcal{H}$. The more complex $\mathcal{H}$ is, the smaller the loss of the best function in $\mathcal{H}$ and thus the harder the learner's benchmark. This complexity can be captured by the notion of *sequential Rademacher complexity* introduced by Rakhlin et al. [2010]. Let $\mathcal{H}$ be a set of functions from $\mathcal{X}$ to $\mathbb{R}$. The sequential Rademacher complexity of a hypothesis $\mathcal{H}$ is denoted by $\mathfrak{R}_T^{\mathrm{seq}}(\mathcal{H})$ and defined by

$$\mathfrak{R}_T^{\mathrm{seq}}(\mathcal{H}) = \frac{1}{T} \sup_{\mathbf{x}} \mathbb{E}\left[\sup_{h\in\mathcal{H}} \sum_{t=1}^{T} \sigma_t h(x_t(\boldsymbol{\sigma}))\right], \tag{4.1}$$

where the supremum is taken over all $\mathcal{X}$-*valued complete binary trees* of depth T and where $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_T)$ is a sequence of i.i.d. Rademacher variables, each taking values in $\{\pm 1\}$ with probability $\frac{1}{2}$. Here, an $\mathcal{X}$-valued complete binary tree $\mathbf{x}$ is defined as a sequence $(x_1, \ldots, x_T)$ of mappings where $x_t \colon \{\pm 1\}^{t-1} \to \mathcal{X}$. The root x_1 can be thought of as some constant in $\mathcal{X}$. The left child of the root is $x_2(-1)$ and the right child is $x_2(1)$. A path in the tree is $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_{T-1})$. To simplify the notation, we write $x_t(\boldsymbol{\sigma})$ instead of $x_t(\sigma_1, \ldots, \sigma_{t-1})$. The sequential Rademacher complexity can be interpreted as the online counterpart of the standard Rademacher complexity widely used in the analysis of batch learning [Bartlett and Mendelson, 2002, Koltchinskii and Panchenko, 2002]. It has been used by Rakhlin et al. [2010] and Rakhlin et al. [2012] both to derive attainability results for some regret bounds and to guide the design of new online algorithms.

In particular, we recall the following central result, which, for any on-line learning game, asserts the existence of a randomized player strategy such that the

minimax value of the game is at most twice the sequential Rademacher complexity of the hypothesis class.

Theorem 4.1 (Theorem 2, [Rakhlin et al., 2010]). *The minimax value of a randomized game is bounded as:* $\mathcal{V}_T(\mathcal{F}) \leq 2\mathfrak{R}_T^{\text{seq}}(\mathcal{F})$.

4.2 Theoretical guarantees

4.2.1 Binary classification

In this section, we present learning guarantees for structural online learning in binary classification. Hence, for any $t \in [1, T]$, the loss incurred at each time t by hypothesis h is $L_t(h(x_t)) = 1_{\{y_t h(x_t) < 0\}}$, with $y_t \in \mathcal{Y} = \{\pm 1\}$.

A *randomized player strategy* $\mathbf{p} = (\mathsf{p}_1, \ldots, \mathsf{p}_T)$ for a sequence of length T is a sequence of mappings $\mathsf{p}_t \colon (\mathcal{X} \times \mathcal{Y})^{t-1} \to \mathcal{P}_{\mathcal{H}}$, $t \in [T]$, where $\mathcal{P}_{\mathcal{H}}$ is the family of distributions over $\mathcal{H}$. Thus, $\mathsf{p}_t((x_1, y_1), \ldots, (x_{t-1}, y_{t-1}))$ is the distribution according to which the player selects a hypothesis $h \in \mathcal{H}$ at time t and which also depends on the past sequence $(x_1, y_1), \ldots, (x_{t-1}, y_{t-1})$ played against the adversary.

The following shows the existence of a randomized strategy that benefits from a margin-based regret guarantee in the online setting. This can be viewed as the counterpart of the classical margin-based learning bounds in the batch setting given by Koltchinskii and Panchenko [2002].

Theorem 4.2 (Proposition 25, [Rakhlin et al., 2010]). *For any function class* $\mathcal{H} \subset \mathbb{R}^{\mathcal{X}}$ *bounded by one, there exists a randomized player strategy given by* $\mathbf{p}$ *such that for any sequence* $z_1, \ldots, z_T$ *played by the adversary,* $z_t = (x_t, y_t) \in \mathcal{X} \times \{\pm 1\}$,

the following inequality holds:

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{h_t\sim \mathsf{p}_t}\left[1_{\{y_t h_t(x_t)<0\}}\right]\right] \leq \inf_{\gamma>0}\left\{\inf_{h^*\in\mathcal{H}}\frac{1}{T}\sum_{t=1}^{T}1_{\{y_t h^*(x_t)<\gamma\}} + \frac{4}{\gamma}\mathcal{R}_T^{\mathrm{seq}}(\mathcal{H}) + \frac{1}{\sqrt{T}}\left(3+\log\log\frac{1}{\gamma}\right)\right\}.$$

We are not explicitly indicating the dependency of p_t on $(x_1, y_1), \ldots, (x_{t-1}, y_{t-1})$ to alleviate the notation. The theorem gives a guarantee for the expected error of a randomized strategy in terms of the empirical margin loss and the sequential Rademacher complexity of $\mathcal{H}$ scaled by γ for the best choice of $h \in \mathcal{H}$ and the best confidence margin γ. As with standard margin bounds, this is subject to a trade-off: for a larger γ the empirical margin loss is larger, while a smaller value of γ increases the complexity term. The result gives a very favorable guarantee when there exists a relatively large γ for which the empirical margin loss of the best h is relatively small.

While this result is remarkable for characterizing learnability against the best-in-class hypothesis, it does not identify and take advantage of any structure in the hypothesis set. The structural margin bound that we prove next specifically provides a guarantee that exploits the scenario where the hypothesis set admits a decomposition $\mathcal{H} = \cup_{k=1}^{p}\mathcal{H}_k$. For any $q \in \mathbb{N}$, we will denote by Δ_q the probability simplex in $\mathbb{R}^q$ and by $\mathrm{conv}(\mathcal{H})$ the convex hull of $\mathcal{H}$.

Theorem 4.3 (Structural online learning bound for binary classification)**.** *Let $\mathcal{H} \subset [-1,1]^{\mathcal{X}}$ be a family of functions admitting a decomposition $\mathcal{H} = \bigcup_{k=1}^{p}\mathcal{H}_k$. Then, there exists a randomized player strategy given by $\mathbf{p}$ on $\mathrm{conv}(\mathcal{H})$ such that*

for any sequence $((x_t, y_t))_{t\in[T]}$ *in* $\mathcal{X} \times \{\pm 1\}$*, the following inequality holds:*

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{h_t\sim \mathsf{p}_t}\left[1_{\{y_t f_t(x_t)<0\}}\right]\right] \leq \inf_{\gamma>0}\left\{\inf_{\substack{h^*=\sum_{i=1}^q \alpha_i^* h_i^* \in \mathrm{conv}(\mathcal{H}) \\ \boldsymbol{\alpha}^*\in\Delta_q, h_i^*\in\mathcal{H}_{k(h_i)}}} \frac{1}{T}\sum_{t=1}^{T} 1_{\{y_t h^*(x_t)<\gamma\}} + \frac{6}{\gamma}\sum_{i=1}^{q}\alpha_i^* \mathfrak{R}_T^{\mathrm{seq}}(\mathcal{H}_{k(h_i^*)}) + \widetilde{\mathcal{O}}\left(\frac{1}{\gamma}\sqrt{\frac{\log p}{T}}\right)\right\},$$

where $k(h_i^*) \in [1, p]$ *is defined to be the smallest index* k *such that* $h_i^* \in \mathcal{H}_k$ *and* $q \in \mathbb{N}$ *so that* h^* *is an arbitrary element in the convex hull.*

The theorem extends the margin bound of Theorem 4.2 given for a single hypothesis set to a guarantee for the convex hull of p hypothesis sets. Observe that, remarkably, the complexity term depends on the mixture weights α_i and hypotheses h_i defining the the best-in-class hypothesis $h^* = \sum_{i=1}^q \alpha_i h_i$. The complexity term is an $\boldsymbol{\alpha}$-average of the sequential Rademacher complexities. Thus, the theorem shows the existence of a randomized strategy $\mathbf{p}$ that achieves a favorable guarantee so long as the best-in-class hypothesis $h^* \in \mathrm{conv}(\mathcal{H})$ admits a decomposition for which the complexity term is relatively small, which directly depends on the amount of mixture weight assigned to more complex $\mathcal{H}_k$s versus less complex ones in the decomposition of h^*.

From a proof standpoint, it is enticing to use the fact that the sequential Rademacher complexity of a hypothesis set does not increase upon taking the convex hull. While this property yields an interesting result itself, it is not sufficiently fine for deriving the result of Theorem 4.3: in short, the resulting guarantee is then in terms of the maximum of the sequential Rademacher complexities instead of their $\boldsymbol{\alpha}$-average.

Proof. Fix $n \geq 1$. For any p-tuple of non-negative integers $\mathbf{N} = (N_1, \ldots, N_p) \in \mathbb{N}^p$

with $|\mathbf{N}| = \sum_{k=1}^{p} N_k = n$, consider the following family of functions:

$$\mathcal{F}_{\mathcal{H},\mathbf{N}} = \left\{ \frac{1}{n} \sum_{k=1}^{p} \sum_{j=1}^{N_k} h_{k,j} \;\middle|\; \forall (k,j) \in [1,p] \times [N_k], h_{k,j} \in H_k \right\}.$$

By the sub-additivity of the supremum operator, the sequential Rademacher complexity of $\mathcal{H}$ can be upper bounded as follows:

$$\begin{aligned}
\mathfrak{R}_T^{\text{seq}}(\mathcal{F}_{\mathcal{H},\mathbf{N}}) &= \frac{1}{T} \sup_{\mathbf{x}} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sup_{h \in \mathcal{F}_{\mathcal{H},N}} \sum_{t=1}^{T} \frac{1}{n} \sum_{k=1}^{p} \sum_{j=1}^{N_k} h_{k,j}(x_t(\boldsymbol{\sigma}))\sigma_t \right] \\
&\leq \frac{1}{T} \frac{1}{n} \sum_{k=1}^{p} \sum_{j=1}^{N_k} \sup_{\mathbf{x}} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sup_{h_{k,j} \in \mathcal{H}_k} \sum_{t=1}^{T} h_{k,j}(x_t(\boldsymbol{\sigma}))\sigma_t \right] = \frac{1}{n} \sum_{k=1}^{p} N_k \mathfrak{R}_T^{\text{seq}}(\mathcal{H}_k).
\end{aligned}$$

In view of this inequality and the margin bound of Proposition 4.2, for any $\mathcal{F}_{\mathcal{H},\mathbf{N}}$, there exists a player strategy $\mathbf{p}^{\mathbf{N}}$ such that

$$\begin{aligned}
&\frac{1}{T} \mathbb{E} \left[\sum_{t=1}^{T} \mathop{\mathbb{E}}_{h_t \sim \mathbf{p}_t^{\mathbf{N}}} [1_{\{y_t f_t(x_t) < 0\}}] \right] \\
&\quad \leq \inf_{\gamma > 0} \left\{ \inf_{g \in \mathcal{F}_{\mathcal{H},\mathbf{N}}} \frac{1}{T} \sum_{t=1}^{T} 1_{\{y_t g(x_t) < \gamma\}} + \frac{4}{\gamma} \frac{1}{n} \sum_{k=1}^{p} N_k \mathfrak{R}_T^{\text{seq}}(\mathcal{H}_k) + \frac{3 + \log\log \frac{1}{\gamma}}{\sqrt{T}} \right\}.
\end{aligned}$$

Now, let $\mathbf{p}^{\text{exp}}$ denote a randomized weighted majority strategy $\mathbf{p}^{\text{exp}}$ with the $\mathbf{p}^{\mathbf{N}}$ strategies serving as experts [Littlestone and Warmuth, 1994]. Since there are at most p^n p-tuples $\mathbf{N}$ with $|\mathbf{N}| = n$, the regret of this randomized weighted majority strategy is bounded by $2\sqrt{T \log(p^n)}$ (see [Cesa-Bianchi and Lugosi, 2006]). Thus, the following guarantee holds for the strategy $\mathbf{p}^{\text{exp}}$:

$$\mathbb{E} \left[\sum_{t=1}^{T} \mathop{\mathbb{E}}_{h_t \sim \mathbf{p}_t^{\text{exp}}} [1_{\{y_t h_t(x_t) < 0\}}] \right] \leq \inf_{|\mathbf{N}| = n} \mathbb{E} \left[\sum_{t=1}^{T} \mathop{\mathbb{E}}_{h_t \sim \pi_t^{\mathbf{N}}} [1_{\{y_t h_t(x_t) < 0\}}] \right] + \sqrt{4Tn \log p}.$$

In view of that, we can write

$$\frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T}\mathop{\mathbb{E}}_{h_t\sim \mathsf{p}_t^{\exp}}[1_{\{y_t h_t(x_t)<0\}}]\right] \leq \inf_{\gamma>0}\left\{\inf_{\substack{g\in\mathcal{F}_{\mathcal{H},\mathbf{N}}\\ |\mathbf{N}|=n}}\frac{1}{T}\sum_{t=1}^{T}1_{\{y_t g(x_t)<\gamma\}}\right.$$
$$\left.+\frac{4}{\gamma}\frac{1}{n}\sum_{k=1}^{p}N_k\mathfrak{R}_T^{\mathrm{seq}}(\mathcal{H}_k)+\frac{3+\log\log\frac{1}{\gamma}}{\sqrt{T}}+2\sqrt{\frac{n\log p}{T}}\right\}$$
$$=\inf_{\gamma>0}\left\{\inf_{\substack{g=\frac{1}{n}\sum_{i=1}^{q}n_i h_i\\ h_i\in\mathcal{H}_{k(h_i)}}}\frac{1}{T}\sum_{t=1}^{T}1_{\{y_t g(x_t)<\gamma\}}+\frac{4}{\gamma}\frac{1}{n}\sum_{i=1}^{q}n_i\mathfrak{R}_T^{\mathrm{seq}}(\mathcal{H}_{k(h_i)})\right.$$
$$\left.+\frac{3+\log\log\frac{1}{\gamma}}{\sqrt{T}}+2\sqrt{\frac{n\log p}{T}}\right\}. \tag{4.2}$$

Now, fix $(h_1^*,\ldots,h_q^*)$. Any $\boldsymbol{\alpha}^*\in\Delta_q$ defines a distribution over $h_1^*,\ldots,h_q^*$. Sampling according to $\boldsymbol{\alpha}^*$ and averaging leads to functions g of the form $g=\frac{1}{n}\sum_{i=1}^{q}n_i h_i$ for some q-tuple $\mathbf{n}=(n_1,\ldots,n_q)$ with $|\mathbf{n}|=n$. Let $h^*=\sum_{i=1}^{q}\alpha_i^* h_i^*$ for some $\boldsymbol{\alpha}^*\in\Delta_q$. By the union bound, we can write, for any $\gamma>0$ and (x_t,y_t),

$$\mathop{\mathbb{E}}_{\mathbf{n}\sim\boldsymbol{\alpha}^*}\left[1_{y_t g(x_t)<\gamma}\right]=\mathbb{P}_{\mathbf{n}\sim\boldsymbol{\alpha}}\left[y_t g(x_t)<\gamma\right]=\mathbb{P}_{\mathbf{n}\sim\boldsymbol{\alpha}^*}\left[y_t g(x_t)-y_t h^*(x_t)+y_t h^*(x_t)<\gamma\right]$$
$$\leq\mathbb{P}_{\mathbf{n}\sim\boldsymbol{\alpha}^*}\left[y_t g(x_t)-y_t h^*(x_t)<-\tfrac{\gamma}{2}\right]+\mathbb{P}_{\mathbf{n}\sim\boldsymbol{\alpha}^*}\left[y_t h^*(x_t)<\tfrac{3\gamma}{2}\right]$$
$$=\mathbb{P}_{\mathbf{n}\sim\boldsymbol{\alpha}^*}\left[y_t g(x_t)-y_t h^*(x_t)<-\tfrac{\gamma}{2}\right]+1_{y_t h^*(x_t)<\frac{3\gamma}{2}}.$$

For any $\gamma>0$ and (x_t,y_t), by Hoeffding's inequality, the following holds:

$$\mathbb{P}_{\mathbf{n}\sim\boldsymbol{\alpha}^*}\left[y_t g(x_t)-y_t h^*(x_t)<-\tfrac{\gamma}{2}\right]\leq e^{\frac{-n\gamma^2}{8}}.$$

Plugging this inequality back into the previous one gives:

$$\mathop{\mathbb{E}}_{\mathbf{n}\sim\boldsymbol{\alpha}^*}\left[1_{y_t g(x_t)<\gamma}\right]\leq e^{\frac{-n\gamma^2}{8}}+1_{y_t h^*(x_t)<\frac{3\gamma}{2}}. \tag{4.3}$$

Fix $\gamma > 0$ and $h_1^* \in \mathcal{H}_{k(h_1^*)}, \ldots, h_q^* \in \mathcal{H}_{k(h_q^*)}$ in inequality 4.2. Then, taking the expectation over $\boldsymbol{\alpha}^*$ of both sides of the inequality and using inequality 4.3 combined with $\mathbb{E}[\frac{n_i}{n}] = \alpha_i^*$, we obtain that for any $\gamma > 0$, any $h_1^* \in \mathcal{H}_{k(h_1^*)}, \ldots, h_q^* \in \mathcal{H}_{k(h_q^*)}$, and any $\boldsymbol{\alpha}^* \in \Delta_q$, the following holds for $h^* = \sum_{i=1}^q \alpha_i^* h_i^*$:

$$\frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T} \mathop{\mathbb{E}}_{h_t \sim \mathsf{p}_t^{\exp}}[1_{\{y_t h_t(x_t) < 0\}}]\right] \leq \frac{1}{T}\sum_{t=1}^{T} 1_{y_t h^*(x_t) < \frac{3\gamma}{2}} + \frac{4}{\gamma}\sum_{i=1}^{q} \alpha_i^* \mathfrak{R}_T^{\text{seq}}(\mathcal{H}_{k(h_i^*)}) + e^{\frac{-n\gamma^2}{8}} + \frac{3 + \log\log\frac{1}{\gamma}}{\sqrt{T}} + 2\sqrt{\frac{n \log p}{T}}.$$

Thus, this inequality holds for any $\gamma > 0$, any $n \geq 1$, and any $h^* = \sum_{i=1}^q \alpha_i h_i \in \text{conv}(\mathcal{H})$. Choosing $n = \left\lceil \frac{4}{\gamma^2} \log \frac{\gamma^2 T}{16 \log p} \right\rceil$ and replacing $\frac{3\gamma}{2}$ by γ yields

$$\frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T} \mathop{\mathbb{E}}_{h_t \sim \mathsf{p}_t^{\exp}}[1_{\{y_t h_t(x_t) < 0\}}]\right] \leq \inf_{\gamma > 0} \left\{ \inf_{\substack{h^* = \sum_{i=1}^q \alpha_i^* h_i^* \in \text{conv}(\mathcal{H}) \\ \boldsymbol{\alpha}^* \in \Delta_q, h_i^* \in \mathcal{H}_{k(h_i)}}} \frac{1}{T}\sum_{t=1}^{T} 1_{y_t h^*(x_t) < \gamma} + \frac{6}{\gamma}\sum_{i=1}^{q} \alpha_i^* \mathfrak{R}_T^{\text{seq}}(\mathcal{H}_{k(h_i^*)}) + 6\sqrt{\frac{\log p}{\gamma^2 T}} + 6\sqrt{\left\lceil \frac{1}{\gamma^2} \log \left\lceil \frac{\gamma^2 T}{36 \log p} \right\rceil \right\rceil \frac{\log p}{T}} + \frac{3 + \log\log\frac{3}{2\gamma}}{\sqrt{T}} \right\},$$

which completes the proof. □

4.2.2 Multiclass classification

In this section, we present structural learning guarantees for online learning for multiclass classification. Here, we assume the supervised learning setting where the output space $\mathcal{Y} = \{1, 2 \ldots, c\}$ is a finite set of labels. We adopt the common formulation of hypothesis sets as scoring functions $\mathcal{H} \subseteq \mathbb{R}^{\mathcal{X} \times \mathcal{Y}}$, where $h(x, y)$ denotes the score of hypothesis h on label y given feature x. Prediction is based on the

label with the largest score:

$$\widehat{y}_h(x) = \operatorname*{argmax}_{y \in \mathcal{Y}} h(x, y).$$

An important concept in multiclass classification with scoring functions is that of margin. The margin of $\rho_h(x, y)$ of hypothesis h on a feature-label pair (x, y) is defined as the difference between the hypothesis's score assigned to y and the highest score assigned to any other label: $\rho_h(x, y) = h(x, y) - \max_{\tilde{y} \neq y} h(x, \tilde{y})$. Notice that if we pick hypothesis h_t at time t with feature-label pair (x_t, y_t), then

$$\widehat{y}_{h_t}(x_t) \neq y_t \Leftrightarrow \rho_{h_t}(x_t, y_t) \leq 0.$$

For convenience, we will use the shorthand $\widehat{y}_t$ for $\widehat{y}_{h_t}(x_t)$. The multiclass classification loss can thus be defined by: $L(\widehat{y}_t, y_t) = 1_{\rho_{h_t}(x_t, y_t) \leq 0}$.

As in the binary classification setting, we are interested in deriving learning bounds that are based on a confidence margin. Thus, we also introduce the margin loss: $L_\gamma(\widehat{y}_t, y_t) = 1_{\rho_{h_t}(x_t, y_t) \leq \gamma}$. With these definitions, we first present the following margin guarantee for multiclass online learning:

Theorem 4.4 (Margin bound for online multiclass classification)**.** *Let* $\Pi_1(\mathcal{H}) = \{x \mapsto h(x, y) : y \in \mathcal{Y}, h \in \mathcal{H}\}$. *Then there exists a randomized player strategy* $\mathbf{p}$

such that:

$$\frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{h_t\sim \mathsf{p}_t}\left[1_{\rho_{h_t}(x_t,y_t)\leq 0}\right] \leq \inf_{\gamma>0}\left\{\inf_{h^*\in\mathcal{H}}\frac{1}{T}\sum_{t=1}^{T}1_{\rho_{h^*}(x_t,y_t)\leq\gamma} + \frac{32\big(1+4\sqrt{2}\log(eT^2)^{3/2}\big)}{\gamma}\mathfrak{R}_T^{\mathrm{seq}}\left(\Pi_1(\mathcal{H})\right) + \frac{\left(1+2\log\frac{\log\left(\frac{2}{\gamma}\right)\pi}{\sqrt{6}}\right)}{\sqrt{T}}\right\}.$$

Proof. Consider the following surrogate loss:

$$\tilde{L}_\gamma(z) = \begin{cases} 1 & \text{if } z\leq 0 \\ 1-\frac{z}{\gamma} \text{ if} & 0<z\leq\gamma \\ 0 & \text{if } z>\gamma. \end{cases}$$

By applying Thereom 4.1 to the loss function $\tilde{L}_\gamma$ and the hypothesis class $\tilde{L}_\gamma\circ\rho_{\mathcal{H}} = \{(x,y)\mapsto \tilde{L}_\gamma(\rho_h(x,y)) : h\in\mathcal{H}\}$, it follows that there exists a randomized player strategy $\mathbf{p}^\gamma$ for which

$$\frac{1}{T}\mathop{\mathbb{E}}_{h_t\sim\mathsf{p}_t^\gamma}\left[\sum_{t=1}^{T}\tilde{L}_\gamma(\rho_{h_t}(x_t,y_t))\right] \leq \inf_{h^*\in\mathcal{H}}\frac{1}{T}\sum_{t=1}^{T}\tilde{L}_\gamma(\rho_{h^*}(x_t,y_t)) + 2\mathfrak{R}_T^{\mathrm{seq}}(L_\gamma\circ\rho_{\mathcal{H}}).$$

By construction, we can lower bound the surrogate loss by the multiclass classification loss and upper bound the surrogate loss by the margin loss. We thus obtain

$$\mathop{\mathbb{E}}_{h_t\sim\mathsf{p}_t^\gamma}\left[\sum_{t=1}^{T}1_{\rho_{h_t}(x_t,y_t)\leq 0}\right] \leq \inf_{h^*\in\mathcal{H}}\sum_{t=1}^{T}1_{\rho_{h^*}(x_t,y_t)\leq\gamma} + 2\mathfrak{R}_T^{\mathrm{seq}}(\tilde{L}_\gamma\circ\rho_{\mathcal{H}}).$$

To bound the sequential Rademacher complexity, notice that by construction,

$\tilde{L}_\gamma$ is $\frac{1}{\gamma}$-Lipschitz. Moreover, by defining $\psi : \mathbb{R}^c \times \mathcal{X} \times \mathcal{Y}$ by $\varphi(z_1, \ldots, z_c, x, y) = \sum_{j=1}^c 1_{y=j} \left(z_j - \max_{\tilde{y} \neq j} z_{\tilde{y}}\right)$, for fixed $x, y \in \mathcal{X} \times \mathcal{Y}$, the following holds:

$$
\begin{aligned}
&|\psi(z_1, \ldots, z_c, x, y) - \psi(z'_1, \ldots, z'_c, x, y)| \\
&= \left| \sum_{j=1}^c 1_{y=j} \left(z_j - \max_{\tilde{y} \neq j} z_{\tilde{y}} \right) - \sum_{j=1}^c 1_{y=j} \left(z'_j - \max_{\tilde{y} \neq j} z'_{\tilde{y}} \right) \right| \\
&\leq \left| \sum_{j=1}^c 1_{y=j} (z_j - z'_j) + \sum_{j=1}^c \max_{\tilde{y} \neq j} (z'_{\tilde{y}} - z_{\tilde{y}}) \right| \\
&\leq 2 \max_{\tilde{y} \in \mathcal{Y}} \left| z_{\tilde{y}} - z'_{\tilde{y}} \right|,
\end{aligned}
$$

so that $\psi(\cdot, x, y)$ is 2-Lipschitz with respect to $\|\cdot\|_\infty$.

Now define $\pi_1(\mathcal{H}) = \{x \mapsto h(x, y) \colon y \in \mathcal{Y}, h \in \mathcal{H}\}$, and recall the following lemma, a contraction inequality for the product of function classes.

Lemma 4.1 (Lemma 4, [Rakhlin et al., 2015]). *Let $\mathcal{G} = \mathcal{G}_1 \times \ldots \mathcal{G}_k$ where each $\mathcal{G}_j \subset [-1, 1]^{\mathcal{X}}$. Further, let $\varphi : \mathbb{R}^k \times \mathcal{Z} \to \mathbb{R}$ be such that $\varphi(\cdot, z)$ is L-Lipschitz with respect to $\|\cdot\|_\infty$ for all $z \in \mathcal{Z}$, and let*

$$
\varphi \circ \mathcal{G} = \{z \mapsto \varphi(g_1(z), g_2(z), \ldots, g_k(z), z) : g_j \in \mathcal{G}_j\}.
$$

Then we have

$$
\mathfrak{R}_T^{\text{seq}}(\varphi \circ \mathcal{G}) \leq 8L\big(1 + 4\sqrt{2} \log(eT^2)^{3/2}\big) \sum_{j=1}^k \mathfrak{R}_T(\mathcal{G}_j),
$$

as long as $\mathfrak{R}_T^{\text{seq}}(\mathcal{G}_j) \geq \frac{1}{T}$ for each j.

By applying this lemma to the surrogate loss, we obtain the following inequality:

$$\mathfrak{R}_T^{\text{seq}}(\tilde{L}_\gamma \circ \rho_{\mathcal{H}}) \leq \frac{16\big(1+4\sqrt{2}\log(eT^2)^{3/2}\big)}{\gamma}\mathfrak{R}_T\left(\Pi_1(\mathcal{H})\right).$$

By combining the above results, we have shown that there exists a randomized player strategy such that

$$\frac{1}{T}\mathop{\mathbb{E}}_{h_t\sim \mathsf{p}_t^\gamma}\big[\sum_{t=1}^{T} 1_{\rho_{h_t}(x_t,y_t)\leq 0}\big] \leq \inf_{h^*\in\mathcal{H}}\frac{1}{T}\sum_{t=1}^{T} 1_{\rho_{h^*}(x_t,y_t)\leq \gamma} + \frac{16\big(1+4\sqrt{2}\log(eT^2)^{3/2}\big)}{\gamma}\mathfrak{R}_T^{\text{seq}}\left(\Pi_1(\mathcal{H})\right).$$

To extend this bound to hold uniformly over γ, for every $k \in \mathbf{N}$, define $\gamma_k = 2^{-k}$. By running an exponentially weighted average algorithm over this countable set of experts and initializing the weight of expert $\mathsf{p}_t^{\gamma_k}$ to be $\frac{6}{\pi^2 k^2}$ (so that they sum to 1), we can guarantee that our algorithm will admit $\mathcal{O}\Big(\sqrt{T\log\left((k\pi)^2/6\right)}\Big)$ regret with respect to having chosen expert k instead.

Specifically, we obtain an algorithm p_t which $\forall k \geq 1$:

$$\frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{h_t\sim \mathsf{p}_t}\left[1_{\rho_{h_t}(x_t,y_t)\leq 0}\right] \leq \inf_{h^*\in\mathcal{H}}\frac{1}{T}\sum_{t=1}^{T} 1_{\rho_{h^*}(x_t,y_t)\leq \gamma_k} + \frac{16\big(1+4\sqrt{2}\log(eT^2)^{3/2}\big)}{\gamma_k}\mathfrak{R}_T^{\text{seq}}\left(\Pi_1(\mathcal{H})\right) + \frac{\left(1+2\log\frac{k\pi}{\sqrt{6}}\right)}{\sqrt{T}}.$$

Now for any $\gamma > 0$, define $k(\gamma) = \min\{k \geq 1 \colon \gamma_k \geq \frac{1}{2}\gamma\}$. Then, $\gamma > \gamma_k$ and

$k(\gamma) \leq \log\left(\frac{2}{\gamma}\right)$, and thus, p_t achieves the guarantee:

$$\frac{1}{T}\sum_{t=1}^{T} \mathop{\mathbb{E}}_{h_t \sim \mathsf{p}_t}\left[1_{\rho_{h_t}(x_t,y_t)\leq 0}\right] \leq \inf_{h^* \in \mathcal{H}} \left\{ \frac{1}{T}\sum_{t=1}^{T} 1_{\rho_{h^*}(x_t,y_t)\leq \gamma} + \frac{32\left(1+4\sqrt{2}\log(eT^2)^{3/2}\right)}{\gamma} \mathfrak{R}_T^{\text{seq}}\left(\Pi_1(\mathcal{H})\right) + \frac{1\left(1+2\log\frac{\log\left(\frac{2}{\gamma}\right)\pi}{\sqrt{6}}\right)}{\sqrt{T}} \right\}.$$

Since $\gamma > 0$ was arbitrary, the result follows. □

As in the setting of binary classification, this margin bound highlights a trade-off between empirical performance of the best-in-class hypothesis and complexity of the family of models that the learner is competing against. If the best-inclass hypothesis has a relatively small margin error even for larger values of γ, then there exists an algorithm for which the learner will achieve favorable guarantees, even if the complexity of the model family is relatively large.

At the same time, this margin bound for multiclass classification does not take advantage of any structure in the hypothesis set. Our next result is a structural margin bound for ensembles of hypotheses in the online multiclass classification setting. This bound will admit more favorable guarantees if there exist ensembles convex combinations of hypotheses that admit small margin error for large values of γ but attribute small weight to more complex hypotheses.

Theorem 4.5 (Structural online learning bound for multiclass classification)**.** *Let* $\mathcal{H} = \cup_{k=1}^{p} \mathcal{H}_k$*, and for any* $h \in \mathcal{H}$*, let* $k(h) = \min\{k \geq 1 : h \in \mathcal{H}_k\}$*. Then, there*

exists a randomized algorithm p_t *for which:*

$$\frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{h_t\sim\mathsf{p}_t}\left[1_{\rho_{h_t}(x_t,y_t)\le 0}\right] \le \inf_{\substack{\gamma>0 \\ h^*=\sum_{i=1}^{q}\alpha_i^* h_i^*\in \mathrm{conv}\left(\cup_{k=1}^{p}\mathcal{H}_k\right)}} \left\{\frac{1}{T}\sum_{t=1}^{T}1_{\rho_{h^*}(x_t,y_t)\le\gamma} + \frac{32\big(1+4\sqrt{2}\log(eT^2)^{3/2}\big)}{\gamma}\sum_{i=1}^{q}\alpha_i^*\mathfrak{R}_T^{\mathrm{seq}}\left(\Pi_1(\mathcal{H}_{k(h_i^*)})\right) + \tilde{\mathcal{O}}\left(\sqrt{\frac{\log(p)}{\gamma^2 T}}\right)\right\}.$$

Proof. Fix $n\in\mathbb{N}$. For any p-tuple $\mathbf{N}=(N_1,\ldots,N_p)\in\mathbb{N}^p$, with $|\mathbf{N}|=n$, consider

$$\mathcal{F}_{\mathcal{H},\mathbf{N}} = \left\{\frac{1}{n}\sum_{k=1}^{p}\sum_{j=1}^{N_k}h_{k,j} : h_{k,j}\in\mathcal{H}_k\right\}.$$

Then $\mathfrak{R}_T^{\mathrm{seq}}(\Pi_1(\mathcal{F}_{\mathcal{H},\mathbf{N}}))\le\frac{1}{n}\sum_{k=1}^{p}N_k\mathfrak{R}_T(\Pi_1(\mathcal{H}_k))$. By Theorem 4.4, for each $\mathcal{F}_{\mathcal{H},N}$, there exists a randomized strategy $\mathbf{p}^N$ on $\mathcal{F}_{\mathcal{H},N}$ such that

$$\frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{h_t\sim\mathsf{p}_t^N}[1_{\rho_{h_t}(x_t,y_t)\le 0}] \le \inf_{\gamma>0}\left\{\inf_{h^*\in\mathcal{F}_{\mathcal{H},N}}\frac{1}{T}\sum_{t=1}^{T}1_{\rho_{h^*}(x_t,y_t)} + \frac{32\big(1+4\sqrt{2}\log(eT^2)^{3/2}\big)}{\gamma}\frac{1}{n}\sum_{k=1}^{p}N_k\mathfrak{R}_T^{\mathrm{seq}}\left(\Pi_1(\mathcal{H}_k)\right) + \frac{\left(1+2\log\frac{\log\left(\frac{2}{\gamma}\right)\pi}{\sqrt{6}}\right)}{\sqrt{T}}\right\}.$$

By applying the exponentially weighted average algorithm over all p^n possible choices for N, we obtain a randomized strategy $\mathbf{p}$ for which

$$\frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{h_t\sim\mathsf{p}_t}\left[1_{\rho_{h_t}(x_t,y_t)\le 0}\right] \le \inf_{\mathbf{N}\in\mathbb{N}^p:|\mathbf{N}|=n}\left\{\sum_{t=1}^{T}\mathop{\mathbb{E}}_{h_t\sim\mathsf{p}_t^N}\left[1_{\rho_{h_t}(x_t,y_t)\le 0}\right] + \sqrt{\frac{4n\log(p)}{T}}\right\}$$

$$\le \inf_{\substack{\mathbf{N}\in\mathbb{N}^p:|\mathbf{N}|=n \\ \gamma>0, h^*\in\mathcal{F}_{\mathcal{H},\mathbf{N}}}}\left\{\frac{1}{T}\sum_{t=1}^{T}1_{\rho_{h^*}(x_t,y_t)\le\gamma} + \sqrt{\frac{4n\log(p)}{T}} + \frac{32\big(1+4\sqrt{2}\log(eT^2)^{3/2}\big)}{\gamma}\frac{1}{n}\sum_{k=1}^{p}N_k\mathfrak{R}_T^{\mathrm{seq}}\left(\Pi_1(\mathcal{H}_k)\right) + \frac{\left(1+2\log\frac{\log\left(\frac{2}{\gamma}\right)\pi}{\sqrt{6}}\right)}{\sqrt{T}}\right\}.$$

Now let $h \in \operatorname{conv}\Big(\cup_{k=1}^{p} \mathcal{H}_k \Big)$ such that $h = \sum_{i=1}^{q} \alpha_i h_i,\ h_i \in \mathcal{H}_{k(h_i)}$. Then $(a_1, \ldots, \alpha_q) \in \Delta_q$ defines a distribution. Thus, by sampling from $(\alpha_1, \ldots, \alpha_q)$ n times and letting $\mathbf{n} = (n_1, \ldots, n_q)$ be the multinomial random variables representing the output, we can define $g = \frac{1}{n} \sum_{i=1}^{q} n_i h_i$.

For any $x, y \in \mathcal{X} \times \mathcal{Y}$,

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\mathbf{n} \sim \boldsymbol{\alpha}} [1_{\rho_g(x,y) < 2\gamma}] = \mathop{\mathbb{E}}_{\mathbf{n} \sim \boldsymbol{\alpha}} [1_{(\rho_g - \rho_h + \rho_h)(x,y) < 2\gamma}] \\
&\leq \mathop{\mathbb{E}}_{\mathbf{n} \sim \boldsymbol{\alpha}} [1_{(\rho_g - \rho_h)(x,y) < -3\gamma}] + \mathop{\mathbb{E}}_{\mathbf{n} \sim \boldsymbol{\alpha}} [1_{\rho_h(x,y) < \gamma}] = \mathop{\mathbb{E}}_{\mathbf{n} \sim \boldsymbol{\alpha}} [1_{(\rho_g - \rho_h)(x,y) < -3\gamma}] + 1_{\rho_h(x,y) < \gamma}.
\end{aligned}$$

Moreover, if we let $y_g = \operatorname{argmax}_{\tilde{y} \neq y} g(x, \tilde{y})$, then, we can compute:

$$\begin{aligned}
\mathop{\mathbb{E}}_{\mathbf{n} \sim \boldsymbol{\alpha}} [1_{(\rho_g - \rho_h)(x,y) < -4\gamma}] &= \mathbb{P}[\rho_g(x,y) - \rho_h(x,y) < -3\gamma] \\
&= \mathbb{P}[g(x,y) - \max_{\tilde{y} \neq y} g(x,\tilde{y}) - h(x,y) + \max_{\tilde{y} \neq y} h(x,\tilde{y}) < -3\gamma] \\
&\leq \mathbb{P}[g(x,y) - g(x,y_g) - (h(x,y) - h(x,y_g)) < -3\gamma] \\
&\leq \sum_{\tilde{y} \neq y} \mathbb{P}[g(x,y) - g(x,\tilde{y}) - (h(x,y) - h(x,\tilde{y})) < -3\gamma] \\
&\leq (c-1) e^{\frac{-9n\gamma^2}{2}}.
\end{aligned}$$

Thus, it follows that:

$$\begin{aligned}
\frac{1}{T} \sum_{t=1}^{T} \mathop{\mathbb{E}}_{h_t \sim \mathsf{p}_t} \left[1_{\rho_{h_t}(x_t, y_t) \leq 0} \right] \leq \inf_{\substack{\gamma > 0 \\ h^* = \sum_{i=1}^{q} \alpha_i^* h_i^* \in \operatorname{conv}\left(\cup_{k=1}^{p} \mathcal{H}_k \right)}} \Bigg\{ & \frac{1}{T} \sum_{t=1}^{T} 1_{\rho_{h^*}(x_t, y_t) \leq \gamma} \\
& + \frac{32\big(1 + 4\sqrt{2} \log(eT^2)^{3/2}\big)}{\gamma} \sum_{i=1}^{q} \alpha_i^* \mathfrak{R}_T^{\mathrm{seq}}\Big(\Pi_1(\mathcal{H}_{k(h_i^*)}) \Big) + \frac{1 + 2 \log \frac{\log\left(\frac{2}{\gamma}\right)\pi}{\sqrt{6}}}{\sqrt{T}} \\
& + (c-1) e^{\frac{-9n\gamma^2}{2}} + \sqrt{\frac{4n \log(p)}{T}} \Bigg\}.
\end{aligned}$$

Setting $n = \left\lceil \frac{1}{9\gamma^2} \log\left(\frac{9(c-1)^2\gamma^2 T}{4\log(p)}\right)\right\rceil$ yields

$$\frac{1}{T}\sum_{t=1}^{T} \mathop{\mathbb{E}}_{h_t \sim \mathsf{p}_t}\left[1_{\rho_{h_t}(x_t,y_t)\leq 0}\right] \leq \inf_{\substack{\gamma>0 \\ h^*=\sum_{i=1}^q \alpha_i^* h_i^* \in \mathrm{conv}\left(\cup_{k=1}^p \mathcal{H}_k\right)}} \left\{ \frac{1}{T}\sum_{t=1}^{T} 1_{\rho_{h^*}(x_t,y_t)\leq\gamma} + \frac{32\left(1+4\sqrt{2}\log(eT^2)^{3/2}\right)}{\gamma}\sum_{i=1}^{q}\alpha_i^* \mathfrak{R}_T^{\mathrm{seq}}\left(\Pi_1(\mathcal{H}_{k(h_i^*)})\right) + \frac{1+2\log\frac{\log\left(\frac{2}{\gamma}\right)\pi}{\sqrt{6}}}{\sqrt{T}} + \frac{1}{\gamma}\sqrt{\frac{4\log p}{9T}} + \sqrt{\left\lceil \frac{1}{9\gamma^2}\log\left(\frac{9(c-1)^2\gamma^2 T}{4\log(p)}\right)\right\rceil \frac{4\log(p)}{T}} \right\}.$$

□

This theorem extends the result of Theorem 4.3 to the online multiclass classification setting as well as the result of Theorem 4.4 to the structural ensemble online learning setting. It admits the same remarkable properties as Theorem 4.3 in that it asserts the existence of randomized strategies with favorable guarantees as long as there exists a best-in-class ensemble with both small margin loss and less weight on more complex hypotheses. Moreover, if the best-in-class ensemble is able to achieve small margin loss for larger margin values (i.e. larger γ), then it can also tolerate larger complexity while maintaining a favorable guarantee.

Thus, if a learner is able to design strategies that achieve these types of guarantees, then the learner would be able to choose complex families of hypotheses more liberally without having to worry about poor performance.

4.3 Algorithms for structural online learning

While Theorems 4.3 and 4.5 assert the existence of randomized strategies with favorable structural learning guarantees for online classification, they do not provide

the learner with explicit algorithms. In this section, we give a general algorithm that benefits from the structural online learning bound above.

In what follows, we will fix an arbitrary decomposition of the function class: $\mathcal{H} = \cup_{k=1}^{p} \mathcal{H}_k$. Moreover, we will assume that this decomposition $(\mathcal{H}_k)_{k=1}^{p}$ is *structurally online linear-learnable* in the sense that for any subset $\mathcal{H}_k \subset \mathcal{H}$, $k \in [1, p]$, there exists an online learning algorithm $\mathcal{A}_k$ such that for any time horizon T and every sequence $(x_t, L_t)_{t=1}^{T}$ where L_t is a linear loss function, $\mathcal{A}_k$ selects a sequence of functions $(h_t)_{t=1}^{T}$ satisfying $\sum_{t=1}^{T} L_t(h_t(x_t)) - \min_{h \in \mathcal{H}} \sum_{t=1}^{T} L_t(h(x_t)) = \mathrm{Reg}_{\mathcal{H}_k, T}(\mathcal{A}_k) = o(T)$. For instance, if $\mathcal{H}$ is finite, every decomposition is structurally online linear-learnable, which can be seen by applying a potential-based algorithm [Cesa-Bianchi and Lugosi, 2006]. Note that the notion of structural online linear-learnability is a slight generalization of the concept introduced by Beygelzimer et al. [2015a].

We will also follow in this section the standard method of using a convex surrogate for the zero-one loss function. The will enable us to design algorithms that apply to both binary and multiclass classification, that are deterministic, and that can be used to achieve new structural PAC learning guarantees in the batch setting (the latter will be seen in Section 4.4). Let $\Phi \colon \mathbb{R} \to \mathbb{R}$ be any convex loss function upper bounding the zero-one loss. We further assume that Φ is G-Lipschitz. One standard example is the hinge loss, $\Phi(x) = (1 - x)1_{\{x \leq 1\}}$, which is 1-Lipschitz. Our goal is to design algorithms $\mathcal{A}$ that guarantee structural upper bounds on the following regret term: $\mathrm{Reg}_T(\mathcal{A}) = \max_{h \in \mathrm{conv}(\mathcal{H})} \sum_{t=1}^{T} \Phi(y_t h_t(x_t)) - \Phi(y_t h(x_t))$. For any x, we will denote by $\Phi'(x)$ an arbitrary element of the subgradient of Φ at x.

4.3.1 SOLBoost algorithm

At first glance, the structural learning bound of Theorem 4.3 seems unwieldy since the convex combination of the best-in-class hypothesis is not necessarily well-ordered with respect to the decomposition of the hypothesis set. Moreover, the proof of Theorem 4.3 is based on the existence of online learning algorithms for different subclasses of functions to which a meta-algorithm for learning with *experts* is applied. This is instructive, but it is also computationally infeasible because there are exponentially many experts in the proof.

Addressing the well-ordering issue and the computational problem will be essential to our algorithmic design. Towards the first point, we can *re-organize* the best-in-class hypothesis by writing:

$$\begin{aligned}\sum_{i=1}^{q} \alpha_i \mathfrak{R}_T(\mathcal{H}_{k(h_i)}) &= \sum_{i=1}^{q} \sum_{k=1}^{p} 1_{\{k(h_i)=k\}} \alpha_i \mathfrak{R}_T(\mathcal{H}_{k(h_i)}) \\ &= \sum_{k=1}^{p} \sum_{i=1}^{q} 1_{\{k(h_i)=k\}} \alpha_i \mathfrak{R}_T(\mathcal{H}_k) = \sum_{k=1}^{p} \gamma_k \mathfrak{R}_T(\mathcal{H}_k).\end{aligned}$$

This suggests that learning against the convex hull of a hypothesis class with a structural decomposition can be equivalently cast as learning against each of its individual substructures along with some new set of convex weights. We will use this observation by applying an (efficient) experts-type algorithm to learn these new weights instead of the original weights from the best convex combination.

However, learning against each of the individual substructures proves to be a challenge in and of itself, since typical online learning algorithms, such as the weighted majority algorithm [Littlestone and Warmuth, 1994], are able to provide guarantees against only the single best-in-class hypothesis. Direct application of an

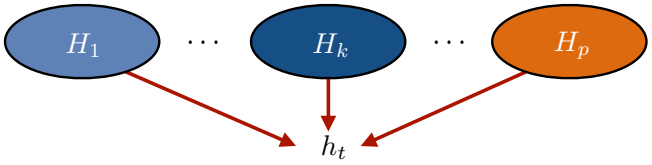


Figure 4.1: Illustration of SOLBOOST Algorithm. The algorithm incorporates a meta-algorithm that measures the progress of each base algorithm. Note from the pseudo-code that the base algorithms are not assigned their true losses, but instead new hallucinated losses.

experts algorithm on top of typical online learning algorithms for each $\mathcal{H}_k$ will only produce a regret guarantee against comparators of the form $\sum_{k=1}^{p} \alpha_k h_k$, $h_k \in \mathcal{H}_k$, $\boldsymbol{\alpha} \in \Delta_p$. On the other hand, Theorem 4.3 guarantees the existence of an algorithm that can attain a structural regret bound against arbitrary convex combinations in $\mathcal{H}$, including those that contain multiple base hypotheses from a single substructure $\mathcal{H}_k$. To attain this type of guarantee, we will linearize the loss and *hallucinate* different losses for each of the base online linear learning algorithms so that they learn well against the convex hull of each subclass.

Our algorithm, SOLBOOST, incorporates these two ideas to produce a guarantee in the form of the one given in Theorem 4.3. Figure 4.1 presents an illustration of the algorithm.

Theorem 4.6. *Let $\mathcal{H} \subset [-1,1]^{\mathcal{X}}$ be a hypothesis set admitting the decomposition $\mathcal{H} = \cup_{k=1}^{p} \mathcal{H}_k$ that is structurally online linear-learnable. For each $k \in [1,p]$, let $\mathcal{A}_k$ be an online algorithm that can minimize the regret of linear loss functions against $\mathcal{H}_k$, with regret $\mathrm{Reg}_{\mathcal{H}_k,T}(\mathcal{A}_k)$ over T rounds. Let $\Phi\colon \mathbb{R} \to \mathbb{R}$ be a G-Lipschitz convex upper bound on the zero-one loss. Then,* SOLBOOST, *initialized with $\eta = \sqrt{\frac{\log p}{G^2 T}}$,*

Algorithm 18: SOLBOOST.

Algorithm: SOLBOOST($(\mathcal{A}_k)_{k=1}^p$, η),
$(\mathcal{A}_k)_{k=1}^p$ structural online linear learning algorithms for the decomposition $\mathcal{H} = \cup_{k=1}^p \mathcal{H}_k$.

$w_{1,k} \leftarrow 1$ for every $k \in [1, p]$
for $t \leftarrow 1, \ldots, T$ **do**
  RECEIVE(x_t)
  for $k \leftarrow 1, \ldots, p$ **do**
    $h_{t,k} \leftarrow$ QUERY($\mathcal{A}_k$)
    $\gamma_{t,k} \leftarrow \frac{w_{t,k}}{\sum_{j=1}^p w_{t,j}}$.
  $h_t \leftarrow \sum_{k=1}^p \gamma_{t,k} h_{t,k}$
  PREDICT($h_t(x_t)$)
  RECEIVE(y_t)
  for $k \leftarrow 1, \ldots, p$ **do**
    ASSIGNLOSS($l_t(y_t h_{t,k}(x_t))$, $\mathcal{A}_k$) where $l_t \colon z \mapsto \Phi'\big(y_t h_t(x_t)\big) y_t z$ is a linear function.
  for $k \leftarrow 1, \ldots, p$ **do**
    $w_{t+1,k} \leftarrow w_{t,k} \left(1 - \eta\, l_t\big(y_t h_{t,k}(x_t)\big)\right)$

outputs a sequence of hypotheses $(h_t)_{t=1}^T$ *that satisfies the following regret bound for any* $(x_t, y_t)_{t=1}^T$*:*

$$\sum_{t=1}^T \Phi(y_t h_t(x_t)) \leq \inf_{\substack{h^* = \sum_{i=1}^q \alpha_i^* h_i^* \in \operatorname{conv}(\mathcal{H}) \\ \boldsymbol{\alpha}^* \in \Delta_q, h_i^* \in \mathcal{H}_{k(h_i^*)}}} \left\{ \sum_{t=1}^T \Phi(y_t h^*(x_t)) + \sum_{i=1}^q \alpha_i^* \operatorname{Reg}_{\mathcal{H}_{k(h_i^*)}, T}(\mathcal{A}_{k(h_i^*)}) + G\sqrt{T \log(p)} \right\}.$$

Proof. Let $\boldsymbol{\alpha}^* \in \Delta_q$ and $(h_i^*)_{i=1}^q \subset \mathcal{H}$ be such that $\sum_{i=1}^q \alpha_i^* h_i^* \in \operatorname{conv}(\mathcal{H})$. For any

$i \in [q]$ and $k \in [1, p]$, define $h^*_{k,i} = 1_{\{k(h^*_i)=k\}} h^*_i$. Then, it follows that

$$\sum_{i=1}^{q} \alpha^*_i h^*_i = \sum_{i=1}^{q} \sum_{k=1}^{p} 1_{\{k(h^*_i)=k\}} \alpha^*_i h^*_i = \sum_{k=1}^{p} \sum_{i=1}^{q} 1_{\{k(h^*_i)=k\}} \alpha^*_i 1_{\{k(h^*_i)=k\}} h^*_i$$
$$= \sum_{k=1}^{p} \left(\sum_{j=1}^{q} 1_{\{k(h^*_j)=k\}} \right) \left[\sum_{i=1}^{q} \frac{1_{\{k(h^*_i)=k\}} \alpha^*_i}{\sum_{j=1}^{q} 1_{\{k(h^*_j)=k\}}} h^*_{k,i} \right] = \sum_{k=1}^{p} \gamma^*_k \sum_{i=1}^{q} \beta^*_{k,i} h^*_{k,i},$$

where $h^*_{k,i} = 1_{\{k(h^*_i)=k\}} h^*_i$, $\gamma^*_k = \sum_{j=1}^{q} 1_{\{k(h^*_j)=k\}}$, $\beta^*_{k,i} = \frac{1_{\{k(h^*_i)=k\}} \alpha^*_i}{\sum_{j=1}^{q} 1_{\{k(h^*_j)=k\}}}$. By convexity of the loss function Φ, we can write

$$\sum_{t=1}^{T} \Phi\left(y_t h_t(x_t)\right) - \Phi\left(y_t \sum_{k=1}^{p} \gamma^*_k \sum_{j=1}^{q} \beta^*_{k,j} h^*_{k,j}(x_t) \right)$$
$$\leq \sum_{t=1}^{T} \sum_{k=1}^{p} \Phi'\left(y_t h_t(x_t)\right) y_t h_{t,k}(x_t) \left[\gamma_{t,k} - \gamma^*_k\right]$$
$$+ \sum_{k=1}^{p} \gamma^*_k \sum_{t=1}^{T} \Phi'\left(y_t h_t(x_t)\right) y_t \left[h_{t,k}(x_t) - \sum_{j=1}^{q} \beta^*_{k,j} h^*_{k,j}(x_t) \right].$$

The first term in the last expression is bounded because the algorithm applies the Prod(η) algorithm [Cesa-Bianchi et al., 2007] to the *hallucinated losses* above. Specifically, we can use the potential function $\log\left(\sum_{k=1}^{p} w_{t,k}\right)$ to track the algo-

rithm's progress against these surrogate losses and compute:

$$\begin{aligned}\log\left(\frac{\sum_{k=1}^{p} w_{t+1,k}}{\sum_{j=1}^{p} w_{1,j}}\right) &= \log\left(\prod_{s=1}^{t}\frac{\sum_{k=1}^{p} w_{s+1,k}}{\sum_{j=1}^{p} w_{s,j}}\right) = \sum_{s=1}^{t}\log\left(\frac{\sum_{k=1}^{p} w_{s+1,k}}{\sum_{j=1}^{p} w_{s,j}}\right)\\ &= \sum_{s=1}^{t}\log\Big(1-\eta\sum_{k=1}^{p}\gamma_{s,k}\Phi'\left(y_t h_t(x_t)\right)y_t h_{t,k}(x_t)\Big)\\ &\leq \sum_{s=1}^{t}-\eta\sum_{k=1}^{p}\gamma_{s,k}\Phi'\left(y_t h_t(x_t)\right)y_t h_{t,k}(x_t),\end{aligned}$$

using the inequality $\log(1+x)\leq x$ for $x\geq -\frac{1}{2}$.

We can also write, for any $k\in[1,p]$,

$$\begin{aligned}\log\left(\frac{\sum_{i=1}^{p} w_{t+1,i}}{\sum_{j=1}^{p} w_{1,j}}\right) &\geq \log\left(\frac{w_{t+1,k}}{\sum_{j=1}^{p} w_{1,j}}\right) = -\log\Big(\sum_{j=1}^{p} w_{1,j}\Big)+\log\left(w_{t+1,k}\right)\\ &\geq -\log\Big(\sum_{j=1}^{p} w_{1,j}\Big)-\sum_{s=1}^{t}\eta\Phi'\left(y_s h_s(x_s)\right)y_s h_{s,k}(x_s)\\ &\quad -\sum_{s=1}^{t}\left(\eta\Phi'\left(y_s h_s(x_s)\right)y_s h_{s,k}(x_s)\right)^2,\end{aligned}$$

in view of $\log(1+x)\geq x-x^2$ for all $x\geq -\frac{1}{2}$, and the constraint on η.

By concavity of the logarithm, this implies that for the $\boldsymbol{\gamma}^*\in\Delta_p$ chosen above,

$$\begin{aligned}\log\left(\frac{\sum_{i=1}^{p} w_{t+1,i}}{\sum_{j=1}^{p} w_{1,j}}\right) &\geq \log\left(\frac{\sum_{k=1}^{p}\gamma_k^* w_{t+1,k}}{\sum_{j=1}^{p} w_{1,j}}\right) \geq \sum_{k=1}^{p}\gamma_k^*\log\left(\frac{w_{t+1,k}}{\sum_{j=1}^{p} w_{1,j}}\right)\\ &\geq \sum_{k=1}^{p}\gamma_k^*\Bigg[-\log\Big(\sum_{j=1}^{p} w_{1,j}\Big)+\sum_{s=1}^{t}-\eta\Phi'\left(y_s h_s(x_s)\right)y_s h_{s,k}(x_s)\\ &\quad -\left(\eta\Phi'\left(y_s h_s(x_s)\right)y_s h_{s,k}(x_s)\right)^2\Bigg].\end{aligned}$$

Combining these calculations yields the inequality:

$$\sum_{t=1}^{T}\sum_{k=1}^{p}\Phi'\Big(y_t h_t(x_t)\Big)y_s h_{t,k}(x_t)(\gamma_{t,k}-\gamma_k^*)$$
$$\leq \sum_{t=1}^{T}\sum_{k=1}^{p}\gamma_k^*\eta\Big(\Phi'\Big(y_t h_t(x_t)\Big)y_s h_{t,k}(x_t)\Big)^2+\frac{1}{\eta}\log(p),$$

since $w_{1,k}=1\ \forall k\in[1,p]$.

For the second term, notice that if for each $k\in[1,p]$ we attribute the loss $l_t(y_t h_{t,k}(x_t))$ to $\mathcal{A}_k$, where l_t is the linear function $l_t\colon z\mapsto\Phi'\big(y_t h_t(x_t)\big)y_t z$, then the fact that l_t is linear implies that $\mathcal{A}_k$ attains some sublinear regret $\mathrm{Reg}_{\mathcal{H}_k,T}(\mathcal{A}_k)$ against $\mathcal{H}_k$: $\max_{h_k^*\in\mathcal{H}_k}\sum_{t=1}^{T}l_t(h_{t,k}(x_t))-\sum_{t=1}^{T}l_t(h_k^*(x_t))\leq\mathrm{Reg}_{\mathcal{H}_k,T}(\mathcal{A}_k)$. Since l_t is a linear loss, this directly implies that the regret guarantee $\mathrm{Reg}_{\mathcal{H}_k,T}(\mathcal{A}_k)$ extends to the convex hull of $\mathcal{H}_k$: $\max\limits_{h_k^*\in\mathrm{conv}(\mathcal{H}_k)}\sum_{t=1}^{T}l_t(h_{t,k}(x_t))-\sum_{t=1}^{T}l_t(h_k^*(x_t))\leq\mathrm{Reg}_{\mathcal{H}_k,T}(\mathcal{A}_k)$. It now follows that:

$$\sum_{t=1}^{T}\Phi\Big(y_t h_t(x_t)\Big)-\Phi\Big(y_t\sum_{k=1}^{p}\gamma_k^*\sum_{j=1}^{q}\beta_{k,j}^*h_{k,j}^*(x_t)\Big)$$
$$\leq\sum_{k=1}^{p}\gamma_k^*\eta\sum_{t=1}^{T}\Big(\Phi'\Big(y_t h_t(x_t)\Big)y_s h_{t,k}(x_t)\Big)^2+\frac{1}{\eta}\log(p)+\sum_{k=1}^{p}\gamma_k^*\mathrm{Reg}_{\mathcal{H}_k,T}(\mathcal{A}_k).$$

Moreover, we can rewrite the convex combination of the regret quantities in terms

of the original convex combination weights of the comparator hypothesis:

$$\begin{aligned}
&\sum_{k=1}^{p} \gamma_k^* \mathrm{Reg}_{\mathcal{H}_k,T}(\mathcal{A}_k) = \sum_{k=1}^{p} \gamma_k^* \sum_{i=1}^{q} \beta_{k,i}^* \mathrm{Reg}_{\mathcal{H}_k,T}(\mathcal{A}_k) \\
&= \sum_{k=1}^{p} \left(\sum_{j=1}^{q} 1_{\{k(h_j^*)=k\}} \right) \left[\sum_{i=1}^{q} \frac{1_{\{k(h_i^*)=k\}} \alpha_i^*}{\sum_{j=1}^{q} 1_{\{k(h_j^*)=k\}}} \mathrm{Reg}_{\mathcal{H}_k,T}(\mathcal{A}_k) \right] \\
&= \sum_{k=1}^{p} \left(\sum_{j=1}^{q} 1_{\{k(h_j^*)=k\}} \right) \left[\sum_{i=1}^{q} \frac{1_{\{k(h_i^*)=k\}} \alpha_i^*}{\sum_{j=1}^{q} 1_{\{k(h_j^*)=k\}}} \mathrm{Reg}_{\mathcal{H}_{k(h_i^*)},T}(\mathcal{A}_{k(h_i^*)}) \right] \\
&= \sum_{k=1}^{p} \sum_{i=1}^{q} 1_{\{k(h_i^*)=k\}} \alpha_i^* \mathrm{Reg}_{\mathcal{H}_{k(h_i^*)},T}(\mathcal{A}_{k(h_i^*)}) = \sum_{i=1}^{q} \alpha_i^* \mathrm{Reg}_{\mathcal{H}_{k(h_i^*)},T}(\mathcal{A}_{k(h_i^*)}).
\end{aligned}$$

Choosing $\eta = \sqrt{\frac{\log(p)}{G^2 T}}$ satisfies the conditions and yields the desired result. □

One remarkable aspect of SOLBoost is that it does not require knowledge of $\mathfrak{R}_T^{\mathrm{seq}}(\mathcal{H}_k)$ for any $\mathcal{H}_k$. As can be seen in Theorem 4.6, these complexity terms are replaced by the regret of each algorithm and are attained automatically. This is a significant advantage over the structural ensemble algorithms in the batch setting (e.g. DeepBoost [Cortes et al., 2014]), which require the learner to either compute or estimate these quantities.

Note that the SOLBoost algorithm is also fundamentally different from applying an online learning algorithm to the DeepBoost objective. The latter would involve calculating the sequential Rademacher complexity for each base hypothesis class and would minimize the cumulative loss of the learner plus a convex combination of sequential Radermacher complexities based on the components in the ensemble. However, this convex combination is not a quantity that the learner seeks to minimize in the online learning. Thus, SOLBoost is applying an online solution to the empirical risk component of the DeepBoost objective.

The bound accompanied by SOLBoost can vastly improve upon bounds that

ignore the structural decomposition. The former realizes an average of all the regrets, and the latter is based upon the maximum regret among all base algorithms.

SOLBoost updates all p base algorithms at each step. To improve the per-round computational cost, we can sample and query only a single base algorithm using techniques from the bandit literature (see e.g. [Cesa-Bianchi and Lugosi, 2006]). This will come at the price of an extra $\sqrt{p}$ factor in the last term on the right-hand side of Theorem 4.6.

SOLBoost can also be compared with recent work on online boosting presented by [Beygelzimer et al., 2015b] and [Beygelzimer et al., 2015a]. In particular, the latter paper interprets online boosting as the task of converting an online learning algorithm with good regret against a base learner into an algorithm with good regret against convex or linear combinations of base learners. This is similar to the type of guarantee that we present.

The work of [Jin et al., 2010] may also seem related to the problem we consider, but it is in fact quite different since the hypothesis sets for online multiple kernel learning and online ensemble learning are distinct. Moreover, the guarantees procured there do not admit arbitrary structural decompositions as in Theorem 4.6 and are only in terms of the best base algorithm.

4.4 Online-to-batch conversion

In this section, we design an effective online-to-batch conversion technique for structural learning. Here, we assume that the learner receives a sample $((x_t, y_t))_{t=1}^T$ in $\mathcal{X} \times \mathcal{Y}$ drawn i.i.d. according to a distribution $\mathcal{D}$. As in previous sections, we assume that the hypothesis space can be decomposed into p subspaces $\mathcal{H} = \cup_{k=1}^p \mathcal{H}_k$.

Algorithm 19: StructuralOTB

Algorithm: StructuralOTB($(\mathcal{A}_k)_{k=1}^p$, $\mathcal{I}$),
$(\mathcal{A}_k)_{k=1}^p$ online learning algorithms for the decomposition $\mathcal{H} = \cup_{k=1}^p \mathcal{H}_k$, $\mathcal{I}$ family of contiguous subintervals in $[1, T]$.

for $t \leftarrow 1, \ldots, T$ **do**
  Receive(x_t)
  $h_{t,k} \leftarrow$ Query($\mathcal{A}_k$)
  Receive(y_t)
  $k_t \leftarrow \operatorname{argmin}_{k \in [1,p]} L(h_{t,k}(x_t), y_t)$

$J_{\text{out}} \leftarrow \operatorname{argmin}_{J \in \mathcal{I}} \frac{1}{|J|} \sum_{t \in J} L(h_{t,k_t}(x_t), y_t) + \sqrt{\frac{2C^2 \log(|\mathcal{I}|/\delta)}{|J|^{1/2}}}$

$h_{J_{\text{out}}} \leftarrow \frac{1}{|J_{\text{out}}|} \sum_{t \in J_{\text{out}}} h_{t,k_t}$

Moreover, we also assume that the learner has access to p base online learning algorithms $(\mathcal{A}_k)_{k=1}^p$, one for each subspace. The objective of the learner is to design a hypothesis h using the online learning algorithms that admits a favorable generalization error $R(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[L(x_t, y_t)]$.

We show that one can design algorithms whose generalization bounds depend on the decomposition of the hypothesis set and the base online learning algorithms. Moreover, these are the first known structural generalization bounds in the batch setting that are in terms of the best-in-class hypothesis and can be viewed as an estimation error extension of the theoretical bounds in [Cortes et al., 2014].

We consider a single static loss function $L\colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ and assume that the difference between rounds is simply the label: $L_t(h_t(x_t)) = L(h_t(x_t), y_t)$. For convenience, we also assume that L is convex in its first argument. Note that any convex surrogate used in Section 4.3 will satisfy this.

Our online-to-batch conversion algorithm, StructuralOTB, is data-dependent and can be viewed as a structural extension of the method of Dekel and Singer [2005]. At a high level, each online learning algorithm $\mathcal{A}_k$ will output

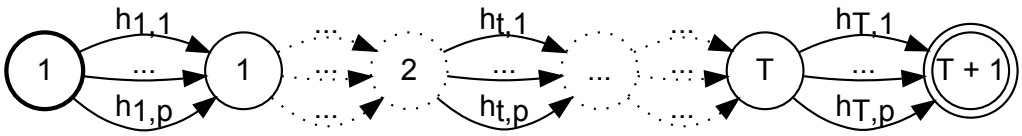


Figure 4.2: Illustration of StructuralOTB. The algorithm chooses the best a posteriori sub-path of hypotheses among all algorithms.

a sequence of hypothesis functions $(h_{t,k})_{t=1}^{T}$. These sequences generate different paths of hypotheses, as illustrated in Figure 4.2. In addition to the base online learning algorithms, our online-to-batch conversion will also take as input a family of contiguous intervals in $[1, T]$. For each interval, our algorithm will find the subpath of hypotheses that has the smallest empirical loss. Selecting a subpath in this way ensures that the subpath will have small empirical estimation error. From these subpaths of hypotheses, the algorithm will select the subpath that has the smallest *penalized* cumulative loss, where shorter paths will have a great additive penalty than longer paths. This is to compensate for the fact that shorter paths may not have seen enough data to generalize well.

Theorem 4.7 presents the guarantee of StructuralOTB. Note that the idea of selecting subpaths for each interval based on the smallest cumulative loss is not immediately obvious or natural, since greedy optimization of empirical error often leads to overfitting and poor generalization.

Theorem 4.7. *Let $((x_t, y_t))_{t=1}^{T}$ be an i.i.d. sample drawn from a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. Assume that the loss function $L \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is bounded by a constant C and that each base online algorithm in the input of* StructuralOTB *admits the following regret guarantee: for any $k \in [1, p]$, $h_k^* \in \mathcal{H}_k$, and contiguous subset $J \subset [1, T]$: $\sum_{t \in J} L(h_{t,k}(x_t), y_t) - L(h_k^*(x_t), y_t) \leq \mathrm{Reg}_J(\mathcal{A}_k)$. Then, with probability*

at least $1-\delta$*, each of the following guarantees holds for* StructuralOTB*:*

$$\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[L(h_{J_{out}}(x),y)\right] \leq \min_{J^*\in\mathcal{I},\alpha^*\in\Delta_p,h_k^*\in\mathcal{H}_k}\left\{\sum_{k=1}^{p}\alpha_k^*\frac{1}{|J^*|}\sum_{t\in J^*}L(h_k^*(x_t),y_t)\right.$$
$$\left.+\sum_{k=1}^{p}\alpha_k^*\mathrm{Reg}_{J^*}(\mathcal{A}_k)+\sqrt{\frac{2C^2\log(|\mathcal{I}|/\delta)}{|J^*|}}\right\},$$
$$\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[L(h_{J_{out}}(x),y)\right] \leq \min_{J^*\in\mathcal{I},\alpha^*\in\Delta_p,h_k^*\in\mathcal{H}_k}\left\{\sum_{k=1}^{p}\alpha_k^*\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[L(h_k^*(x),y)\right]\right.$$
$$\left.+\sum_{k=1}^{p}\alpha_k^*\mathrm{Reg}_{J^*}(\mathcal{A}_k)+\sqrt{\frac{2C^2\log(p/\delta)}{T}}+\sqrt{\frac{2C^2\log(|\mathcal{I}|/\delta)}{|J^*|}}\right\}.$$

Proof. Let $J\subset[1,T]$ be any subset, and denote $h_J=\frac{1}{|J|}\sum_{t\in J}h_{t,k_t}$. Notice that by convexity of L in the first coordinate,

$$\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[L(h_J(x),y)\right]\leq\frac{1}{|J|}\sum_{t\in J}\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[L(h_{t,k_t}(x),y)\right].$$

Now for any $t\in J$, define $M_t=\frac{1}{|J|}\left(L(h_{t,k_t}(\cdot),\cdot)-\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[L(h_{t,k_t}(x),y)\right]\right)$. By design, $(M_t)_{t\in J}$ is a sequence of martingale differences over J, such that if we reindex $J=[1,T_J]$, then $\mathbb{E}[M_s|M_1,\ldots,M_{s-1}]=0$ for every $s\in[1,T_J]$.

Furthermore, by Azuma's inequality, we can guarantee that with probability at least $1-\delta$, $\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[L(h_{t,k_t}(x),y)\right]\leq\frac{1}{|J|}\sum_{t\in J}L(h_{t,k_t}(x_t),y_t)+\sqrt{\frac{2C^2\log(1/\delta)}{|J|}}$. By applying a union bound over all $J\in\mathcal{I}$, then with probability at least $1-\delta$, the following bound holds for every $J\in\mathcal{I}$:

$$\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[L(h_{t,k_t}(x),y)\right]\leq\frac{1}{|J|}\sum_{t\in J}L(h_{t,k_t}(x_t),y_t)+\sqrt{\frac{2C^2\log(|\mathcal{I}|/\delta)}{|J|}}.$$

Thus, it follows that

$$\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[L(h_{J_{\text{out}}}(x),y)\right] \leq \min_{J^*\in\mathcal{I}} \frac{1}{|J^*|}\sum_{t\in J^*} L(h_{t,k_t}(x_t),y_t) + \sqrt{\frac{2C^2\log(|\mathcal{I}|/\delta)}{|J^*|}}.$$

By the choice of k_t, we can further say that $L(h_{t,k_t}(x_t),y_t) \leq L(h_{t,k}(x_t),y_t)\ \forall k \in [1,p]$. In particular, this means that for any $J \in \mathcal{I}$,

$$\begin{aligned}\frac{1}{|J|}\sum_{t\in J} L(h_{t,k_t}(x_t),y_t) &\leq \min_{\alpha^*\in\Delta_p} \frac{1}{|J|}\sum_{t\in J}\sum_{k=1}^{p}\alpha_k^* L(h_{t,k}(x_t),y_t)\\ &\leq \min_{\alpha^*\in\Delta_p, h_k^*\in\mathcal{H}_k} \sum_{k=1}^{p}\alpha_k^*\left(\frac{1}{|J|}\sum_{t\in J} L(h_k^*(x_t),y_t) + \text{Reg}_J(\mathcal{A}_k)\right).\end{aligned}$$

Combining the above two inequalities yields the first result.

Furthermore, we can use Hoeffding's inequality over the best-in-class classifier's guarantee for each of the p subclasses and apply a union bound to say that

$$\begin{aligned}\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[L(h_{J_{\text{out}}}(x),y)\right] \leq \min_{J^*\in\mathcal{I},\alpha^*\in\Delta_p,h_k^*\in\mathcal{H}_k}\Bigg\{&\sum_{k=1}^{p}\alpha_k^*\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[L(h_k^*(x),y)\right]\\ &+\sum_{k=1}^{p}\alpha_k^*\text{Reg}_{J^*}(\mathcal{A}_k) + \sqrt{\frac{2C^2\log(p/\delta)}{T}} + \sqrt{\frac{2C^2\log(|\mathcal{I}|/\delta)}{|J^*|}}\Bigg\}.\end{aligned}$$

□

Note that one natural choice of $\mathcal{I}$, as discussed by Dekel and Singer [2005], is the set of all suffixes of $[1,T]$: $\{[1,T],[2,T],\ldots,[T,T]\}$. This was shown empirically to outperform the "data-independent" online-to-batch conversion methods of Cesa-Bianchi et al. [2004]. With this specific choice of $\mathcal{I}$, $|\mathcal{I}| = T$, and the logarithmic dependence on $|\mathcal{I}|$ is mild.

4.5 Summary of chapter

We presented a series of theoretical and algorithmic results for structural online learning. Our theory and algorithms can be further extended to cover other learning settings beyond classification, including regression, structured prediction, and general online learning. In contrast with the batch algorithms for structural learning, our algorithms do not require the estimation of the Rademacher complexities in the decomposition of the hypothesis set. Moreover, our online-to-batch conversion algorithm provides an efficient alternative to the current structural ensemble methods used in the batch setting.

Conclusion

In this thesis, we presented an extensive study of both dynamic and adaptive online learning.

In particular, we introduced a novel and flexible framework for online learning against dynamic competitors by representing sequences of experts with weighted finite automata. In the process, we not only recovered existing extensions of regret, such as tracking regret, as special cases of our framework, but we also recovered existing algorithms, such as Fixed-Share, using the general concept of automata approximation with n-gram language models. Moreover, we presented methods for compressing automata using φ-transitions, and we designed new online learning algorithms that are able to learn against φ-automata. To accomplish these goals, we extended classical automata algorithms such as composition and shortest-distance to the setting of φ-automata, which may be of independent interest. This framework for online learning allows the learner to design computationally efficient algorithms against arbitrary families of expert sequences, and it both unifies and generalizes existing work in this area.

In addition to competing against fixed expert sequences, we also studied online learning algorithms that are robust under perturbations. We introduced the notion

of n-gram conditional swap regret, which is the regret of a learner's sequence of actions against an arbitrary n-gram finite state transducer. We also showed how this framework generalizes existing work on swap regret, which can be interpreted as unigram swap regret. We presented efficient algorithms that achieve sublinear n-gram conditional swap regret, and we showed how bigram conditional swap regret leads to a natural extension of correlated equilibrium in algorthmic game theory.

This thesis also presented new adaptive algorithms for online convex optimization. In particular, it presented a flexible extension of the classical Follow-the-Regularized-Leader algorithm that takes as input subgradient predictions made by the learner, and adaptively tunes the aggressiveness of its updates based on the accuracy of the learner's predictions. The regret guarantees achieved by this new family of algorithms are data-dependent and can be much more favorable than traditional worst-case guarantees, or even previously designed adaptive algorithms. Our framework can also handle stochastic subgradients and composite terms, making it a suitable upgrade from standard machine learning algorithms such as stochastic gradient descent.

Finally, this thesis also presented a new theoretical and algorithmic treatment for model selection in online learning. We derived *voted risk minimizaion*-type guarantees in the online learning setting and presented margin bounds for online binary and multiclass classification with ensembles of hypotheses. These guarantees leverage the *structure* of the hypothesis space. We also presented an online learning algorithm that captures the structural benefits of our theoretical guarantees, along with an online-to-batch conversion algorithm that allows the learner to design a batch hypothesis with favorable generalization properties.

Online learning has become one of the most important areas of machine learning

due to its flexibility, efficiency, and scalability. This thesis focused on developing and extending online learning in two critical areas: in dynamic environments and with adaptive data-dependent guarantees. In some sense, these two problems were treated independently. It is interesting to ask whether one can design adaptive algorithms that also perform well in dynamic environments, and, more importantly, whether there is a general unifying framework for designing such methods.

Appendix A

Supplementary material for Chapter 1

A.1 Additional proofs

A.1.1 Proof of Theorem 1.1

Proof. We will use a standard potential function-based argument. Let $\tilde{v}_{t,j}$ be the path weights at each iteration computed by the algorithm, and let Φ_t be the potential defined by: $\Phi_t = \log\left(\sum_{j=1}^{K} \tilde{v}_{t,j}\right)$. Then by Hoeffding's inequality,

$$\Phi_t - \Phi_{t-1} = \log\left(\frac{\sum_{j=1}^{K} e^{-\eta l_t[\pi_{j,t}]}\tilde{v}_{t-1,j}}{\sum_{j=1}^{K}\tilde{v}_{t-1,j}}\right) = \log\left(\mathop{\mathbb{E}}_{j\sim\tilde{p}_t}\left[e^{-\eta l_t[\pi_{j,t}]}\right]\right)$$
$$\leq \mathop{\mathbb{E}}_{j\sim\tilde{p}_t}\left[(-\eta)l_t[\pi_{j,t}]\right] + \frac{\eta^2}{8} = \mathop{\mathbb{E}}_{i\sim\mathsf{p}_t}\left[(-\eta)l_t[i]\right] + \frac{\eta^2}{8}.$$

Summing over t results in the following upper bound:

$$\Phi_T - \Phi_1 \leq \sum_{t=1}^{T} \mathop{\mathbb{E}}_{i \sim \mathsf{p}_t} [(-\eta) l_t[i]] + \frac{\eta^2 T}{8}.$$

We can also straightforwardly derive a lower bound for the same quantity. For any $i^* \in [K]$,

$$\begin{aligned}\Phi_T - \Phi_1 &= \log\left(\sum_{j=1}^{K} \tilde{v}_{T,j}\right) - \log\left(\sum_{j=1}^{K} \tilde{v}_{1,j}\right) \geq \log\left(\tilde{v}_{T,i^*}\right) - \log\left(\sum_{j=1}^{K} \tilde{v}_{1,j}\right) \\ &= -\eta \sum_{t=1}^{T} l_t[\pi_{i^*,t}] + \eta \log(\tilde{v}_{1,i^*}) - \log\left(\sum_{j=1}^{K} \tilde{v}_{1,j}\right).\end{aligned}$$

Combining the two statements above yields

$$-\eta \sum_{t=1}^{T} l_t[\pi_{i^*,t}] + \eta \log(\tilde{v}_{1,i^*}) - \log\left(\sum_{j=1}^{K} \tilde{v}_{1,j}\right) \leq \mathop{\mathbb{E}}_{i \sim \mathsf{p}_t} [(-\eta) l_t[i]] + \frac{\eta^2 T}{8},$$

which can be rearranged as

$$\sum_{t=1}^{T} \mathop{\mathbb{E}}_{i \sim \mathsf{p}_t} [l_t[i]] - \sum_{t=1}^{T} l_t[\pi_{i^*,t}] \leq \frac{\eta T}{8} + \frac{1}{\eta} \log\left(\frac{1}{\tilde{v}_{1,i^*}}\right) + \frac{1}{\eta} \log\left(\sum_{j=1}^{K} \tilde{v}_{1,j}\right).$$

If $\tilde{v}_{1,j} = 1$ for all $j \in [1, K]$, then the first result can be computed in a straightforward manner.

If $\tilde{v}_{1,j} = w[\pi_j]^\eta$, then it follows that

$$\sum_{t=1}^{T} \mathop{\mathbb{E}}_{i \sim \mathsf{p}_t} [l_t[i]] - \sum_{t=1}^{T} l_t[\pi_{i^*,t}] \leq \frac{\eta T}{8} + \log\left(\frac{1}{w[\pi_{i^*}]}\right) + \frac{1}{\eta} \log\left(\sum_{j=1}^{K} w[\pi_j]^\eta\right),$$

which implies that

$$\sum_{t=1}^{T} \mathop{\mathbb{E}}_{i\sim \mathsf{p}_t}[l_t[i]] - \sum_{t=1}^{T} l_t[\pi_{i^*,t}] + \log\left(w[\pi_{i^*}]|\mathcal{C}\cap\mathcal{S}_T|\right) \leq \frac{\eta T}{8} + \frac{1}{\eta}\log\left(\sum_{j=1}^{K} w[\pi_j]^\eta |\mathcal{C}\cap\mathcal{S}_T|^\eta\right).$$

When $\sum_{j=1}^{K} w[\pi_j] = 1$, the left-hand side is the weighted regret.

Moreover, maximization of the last term can be written as follows in terms of the Lagrange function:

$$\max_{w} \frac{1}{\eta}\log\left(\sum_{j=1}^{K} w[\pi_j]^\eta\right) + \lambda \sum_{j=1}^{K} w[\pi_j].$$

The partial derivatives of the Lagrangian with respect to w_j must satisfy

$$\frac{1}{\eta}\frac{1}{\sum_{j=1}^{K} w_j^\eta}\eta w_j^{\eta-1} + \lambda = 0,$$

at the solution. Thus, the solution verifies that $w_j = \left(-\lambda\eta\sum_{j=1}^{K} w_j^\eta\right)^{\frac{1}{\eta-1}}$, which is uniform in j. Thus, the maximizing value is $w_j = \frac{1}{K}$, yielding the bound: $\eta T + \frac{1}{\eta}\log(|\mathcal{C}\cap\mathcal{S}_T|)$.

By interpreting w as a distribution, we can consider the $H_\eta(w)$, the η-Rényi entropy of w. Recall that $H_\eta(w) = \frac{1}{1-\eta}\log\left(\sum_{j=1}^{K} w[\pi_j]\right)$. Thus, we can write

$$\begin{aligned}\frac{\eta T}{8} + \frac{1}{\eta}\log\left(\sum_{j=1}^{K} w[\pi_j]^\eta |\mathcal{C}\cap\mathcal{S}_T|^\eta\right) &= \frac{\eta T}{8} + \frac{1-\eta}{\eta}H_\eta(w) + \log(|\mathcal{C}\cap\mathcal{S}_T|)\\ &= \frac{\eta T}{8} + \frac{1}{\eta}H_\eta(w) - H_\eta(w) + \log(K).\end{aligned}$$

Since $\eta \mapsto H_\eta(w)$ is a decreasing function (see e.g. [Van Erven and Harremos, 2014]), it follows that $\eta \mapsto \frac{\eta}{\sqrt{H_\eta(w)}}$ is an increasing function that increases at least

linearly. Since we assume that w is supported on more than a single point, we have $H_0(w) > 0$. Thus, for any T, there exists a unique η^* such that $\frac{\eta^*}{\sqrt{H_{\eta^*}(w)}} = \sqrt{\frac{8}{T}}$. Moreover, it is also the case that for every $\eta \leq \eta^*$, $\frac{\eta}{\sqrt{H_\eta(w)}} \leq \sqrt{\frac{8}{T}}$. Thus, we have

$$\begin{aligned}\frac{\eta T}{8} + \frac{1}{\eta}\log\left(\sum_{j=1}^{K} w[\pi_j]^\eta |\mathcal{C} \cap \mathcal{S}_T|^\eta\right) &\leq \inf_{\eta \leq \eta^*} \frac{\eta T}{8} + \frac{1}{\eta}H_\eta(w) - H_\eta(w) + \log(K) \\ &\leq \sqrt{\frac{T H_{\eta^*}(w)}{8}} - H_{\eta^*}(w) + \log(K).\end{aligned}$$

□

A.1.2 Proof of Lemma 1.1

Proof. Using the definition of binary entropy, the following uper bound holds (which is finer than the one obtained using Pinsker's inequality):

$$
\begin{aligned}
&- D\left(\frac{k+(1/2)}{T-1}||\frac{k}{T-1}\right) \\
&= -D\left(\frac{k}{T-1}\left(1+\frac{1}{2k}\right)||\frac{k}{T-1}\right) \\
&= \left(1+\frac{1}{2k}\right)\frac{k}{T-1}\log\frac{\frac{k}{T-1}}{\left(1+\frac{1}{2k}\right)\frac{k}{T-1}} \\
&\quad + \left(1-\left(1+\frac{1}{2k}\right)\frac{k}{T-1}\right)\log\frac{1-\frac{k}{T-1}}{1-\left(1+\frac{1}{2k}\right)\frac{k}{T-1}} \\
&= \left(1+\frac{1}{2k}\right)\frac{k}{T-1}\log\frac{1}{1+\frac{1}{2k}} \\
&\quad + \left(1-\frac{k}{T-1}-\frac{1}{2k}\frac{k}{T-1}\right)\log\left(1+\frac{\frac{1}{2k}\frac{k}{T-1}}{1-\frac{k}{T-1}-\frac{1}{2k}\frac{k}{T-1}}\right) \\
&\leq \left(1+\frac{1}{2k}\right)\frac{k}{T-1}\frac{-\frac{1}{2k}}{1+\frac{1}{4k}} + \left(1-\frac{k}{T-1}-\frac{1}{2k}\frac{k}{T-1}\right)\frac{\frac{1}{2k}\frac{k}{T-1}}{1-\frac{k}{T-1}-\frac{1}{2k}\frac{k}{T-1}} \\
&\quad \left(\text{using } \log(1+x)\geq\frac{x}{1+\frac{x}{2}} \text{ and } \log(1+x)<x\right) \\
&= \frac{1}{2k}\frac{k}{T-1}\left(1-\frac{1+\frac{1}{2k}}{1+\frac{1}{4k}}\right) = \frac{-\frac{1}{8k^2}\frac{k}{T-1}}{1+\frac{1}{4k}} = \frac{-\frac{1}{4k^2}\frac{k}{T-1}}{2+\frac{1}{4k}} \leq -\frac{1}{12k(T-1)}.
\end{aligned}
$$

Similarly, we can also write:

$$
\begin{aligned}
&- D\left(\left(1-\frac{1}{2k}\right)\frac{k}{T-1}\|\frac{k}{T-1}\right)\\
&=\left(1-\frac{1}{2k}\right)\frac{k}{T-1}\log\frac{\frac{k}{T-1}}{\left(1-\frac{1}{2k}\right)\frac{k}{T-1}}\\
&\quad+\left(1-\left(1-\frac{1}{2k}\right)\frac{k}{T-1}\right)\log\frac{1-\frac{k}{T-1}}{1-\left(1-\frac{1}{2k}\right)\frac{k}{T-1}}\\
&=\left(1-\frac{1}{2k}\right)\frac{k}{T-1}\log\frac{1}{1-\frac{1}{2k}}\\
&\quad+\left(1-\frac{k}{T-1}+\frac{1}{2k}\frac{k}{T-1}\right)\log\left(1-\frac{\frac{1}{2k}\frac{k}{T-1}}{1-\frac{k}{T-1}+\frac{1}{2k}\frac{k}{T-1}}\right)\\
&\leq\left(\frac{1}{2k}-\frac{1}{8k^2}\right)\frac{k}{T-1}+\left(1-\frac{k}{T-1}+\frac{1}{2k}\frac{k}{T-1}\right)\frac{-\frac{1}{2k}\frac{k}{T-1}}{1-\frac{k}{T-1}+\frac{1}{2k}\frac{k}{T-1}}\\
&=-\frac{\frac{1}{4k^2}\frac{k}{T-1}}{2}=-\frac{1}{8k(T-1)}.
\end{aligned}
$$

□

A.1.3 Proof of Theorem 1.5

Proof. Consider the mirror map $\psi\colon(\Delta_N)^m\to\mathbb{R}$ defined by

$$\psi(\mathbf{p})=\sum_{j=1}^{m}\sum_{i=1}^{N}\mathsf{p}_j(i)\log\mathsf{p}_j(i).$$

This induces the Bregman divergence:

$$B_\psi(\mathbf{p},\mathbf{q})=\sum_{j=1}^{m}\sum_{i=1}^{N}\mathsf{p}_j(i)\log\left(\frac{\mathsf{p}_j(i)}{\mathsf{q}_j(i)}\right).$$

Since each relative entropy is 1-strongly convex with respect to the l_1 norm over a single simplex, the additivity of strong convexity implies that B_ψ is 1-strongly convex with respect to the l_1 norm defined over$(\Delta_N)^m$.

The update described in the theorem statement corresponds to the mirror descent update based on B_ψ:

$$\mathbf{p}_{s+1} = \operatorname*{argmin}_{\mathbf{p}\in(\Delta_N)^m} \langle g_s, \mathbf{p}\rangle + B_\psi(\mathbf{p}, \mathbf{p}_s).$$

where $g_s \in \partial(f(\mathbf{p}_s))$ is an element of the subgradient of f at $\mathbf{p}_s$. Thus, the standard mirror descent regret bound (e.g. [Bubeck et al., 2015]) implies that

$$\sum_{s=1}^{\tau} f(\mathbf{p}_s) - f(\mathbf{p}^*) \leq \frac{1}{\eta\tau} \sup_{\mathbf{p}\in(\Delta_N)^m} \psi(\mathbf{p}) - \psi(\mathbf{p}_1) + \eta 2L.$$

The result now follows from the fact that $\psi(\mathbf{p}) \leq 0$ and $\psi(\mathbf{p}_1) = m\log(N)$. □

A.1.4 Proof of Theorem 1.14

Proof. Given any $\mathsf{p}, \mathsf{q} \in \Delta_K$, let $D(\mathsf{p}||\mathsf{q})$ denote the relative entropy between p and q. Given $A_t \subseteq \Sigma$, let $\mathsf{p}(A_t) = \sum_{i\in\{1,\ldots,K\}:\, \pi_{i,t}\in A_t} \mathsf{p}[\pi_{i,t}]$.

We first claim that if for any path π_j, $\pi_{j,t} \notin A_t$, then $\tilde{\mathsf{p}}_{t+1,j} = \tilde{\mathsf{p}}_{t,j}$. This is because $\pi_{j,t} \notin A_t$ implies that $w_{t+1,j} = w_{t,j}$. Moreover, the normalization step guarantees that $\sum_{i\notin A_t} \tilde{\mathsf{p}}_{t+1,i} = \sum_{i\notin A_t} \tilde{\mathsf{p}}_{t,i}$.

Then, for any distribution $u \in \Delta_K$, we can write

$$
\begin{aligned}
&D(u||\tilde{\mathsf{p}}_t) - D(u||\tilde{\mathsf{p}}_{t+1}) \\
&= \sum_{i=1}^{K} u_i \log \frac{\tilde{\mathsf{p}}_{t+1,i}}{\tilde{\mathsf{p}}_{t,i}} \\
&= \sum_{i \in \{1,\dots,K\}\colon \pi_{i,t} \in A_t} u_i \log \frac{\tilde{\mathsf{p}}_{t+1,i}}{\tilde{\mathsf{p}}_{t,i}} \\
&= \sum_{i \in \{1,\dots,K\}\colon \pi_{i,t} \in A_t} u_i \log \frac{\tilde{\mathsf{p}}^{A_t}_{t+1,i}}{\tilde{\mathsf{p}}^{A_t}_{t,i}} \\
&= \sum_{i \in \{1,\dots,K\}\colon \pi_{i,t} \in A_t} u_i \log \frac{\tilde{\mathsf{p}}^{A_t}_{t,i} e^{-\eta l_t[\pi_i,t]}}{\tilde{\mathsf{p}}^{A_t}_{t,i} \sum_{j=1}^{K} \tilde{\mathsf{p}}^{A_t}_{t,j} e^{-\eta l_t[\pi_{j,t}]}} \\
&= \sum_{i \in \{1,\dots,K\}\colon \pi_{i,t} \in A_t} u_i(-\eta l_t[\pi_{i,t}]) - \sum_{i \in \{1,\dots,K\}\colon \pi_{i,t} \in A_t} u_i \log \sum_{j=1}^{K} \tilde{\mathsf{p}}^{A_t}_{t,j} e^{-\eta l_t[\pi_{j,t}]} \\
&= \sum_{i \in \{1,\dots,K\}\colon \pi_{i,t} \in A_t} -\eta u_i l_t[\pi_{i,t}]) - \sum_{i \in \{1,\dots,K\}\colon \pi_{i,t} \in A_t} u_i \log \mathop{\mathbb{E}}_{j \sim \tilde{\mathsf{p}}^{A_t}_t} \left[e^{-\eta l_t[\pi_{j,t}]} \right].
\end{aligned}
$$

By Hoeffding's inequality, we can bound the last term by:

$$
\begin{aligned}
&\geq \sum_{i \in \{1,\dots,K\}\colon \pi_{i,t} \in A_t} -\eta\, u_i\, l_t[\pi_{i,t}]) - \sum_{i \in \{1,\dots,K\}\colon \pi_{i,t} \in A_t} u_i \left[\mathop{\mathbb{E}}_{j \sim \tilde{\mathsf{p}}^{A_t}_t} [-\eta l_t[\pi_{j,t}]] + \frac{\eta^2}{8} \right] \\
&= -\eta \sum_{i \in \{1,\dots,K\}\colon \pi_{i,t} \in A_t} u_i l_t[\pi_{i,t}] + \eta u(A_t) \mathop{\mathbb{E}}_{j \sim \tilde{\mathsf{p}}^{A_t}_t} [l_t[\pi_{j,t}]] - u(A_t)\frac{\eta^2}{8} \\
&= -\eta \sum_{i \in \{1,\dots,K\}\colon \pi_{i,t} \in A_t} u_i l_t[\pi_{i,t}] + \eta u(A_t) \mathop{\mathbb{E}}_{i \sim \mathsf{p}^{A_t}_t} [l_t[i]] - u(A_t)\frac{\eta^2}{8}.
\end{aligned}
$$

After rearranging terms and summing over t, it follows that

$$
\sum_{t=1}^{T} u(A_t) \mathop{\mathbb{E}}_{i \sim \mathsf{p}^{A_t}_t} [l_t[i]] - \sum_{i \in \{1,\dots,K\}\colon \pi_{i,t} \in A_t} u_i l_t[\pi_{i,t}] \leq \frac{\eta}{8} \sum_{t=1}^{T} u(A_t) + \frac{1}{\eta} D(u||\tilde{\mathsf{p}}_1).
$$

For the unweighted regret, initializing $\tilde{v}_1 = \frac{1}{K}$ implies that $D(u||\tilde{\mathsf{p}}_1) \leq \log(K)$, which completes the argument. $\square$

Appendix B

Supplementary material for Chapter 2

B.1 Proof of Theorem 2.1

Proof. If sublinear regret is impossible in the oblivious case, then it is impossible for any level of memory. Now, for any t, let $j_t^* \in \operatorname{argmin}_j p_j^t$ and define for the adversary the following sequence of loss functions:

$$f_t(x^t) = \begin{cases} 1 & \text{for } x^t \neq j_t^* \\ 0 & \text{for } x^t = j_t^*. \end{cases}$$

Algorithm 20: BigramStateDepCSwap.

Algorithm: BigramStateDepCSwap($(\mathcal{A}_{i,j})_{i,j=1}^N$, $\mathcal{S}$),
$(\mathcal{A}_{i,j})_{i,j=1}^N$ external-regret minimizing online learning algorithms such that $\mathcal{A}_{i,j}$ is the same algorithm for all $(i,j) \in \mathcal{S}$.

$\mathsf{p}_1 \leftarrow \frac{1}{N}$.
$a_0 \leftarrow \textsc{Sample}(\mathsf{p}_1)$.
for $t \leftarrow 1, \dots, T$ **do**
  $a_t \leftarrow \textsc{Sample}(\mathsf{p}_t)$.
  $f_t \leftarrow \textsc{Play}(a_t)$.
  for $i, j \leftarrow 1, \dots, N$ **do**
    if $i, j \in \mathcal{S}$ **then**
      $\textsc{AssignLoss}(\frac{\sum_{j\colon (i,j)\in\mathcal{S}} \mathsf{p}_t[i]\delta_{a_{t-1}}^{t-1}[j] f_t}{|\{j\colon (i,j)\in\mathcal{S}\}|}, \mathcal{A}_{i,j})$.
    if $i, j \in \mathcal{S}^c$ **then**
      $\textsc{AssignLoss}(\mathsf{p}_t[i]\delta_{a_{t-1}}^{t-1}[j] f_t, \mathcal{A}_{i,j})$.
    $Q_i^{t+1,j} \leftarrow \textsc{AlgoStrategy}(\mathcal{A}_{i,j})$.
  $Q^{t+1} \leftarrow \sum_{j=1}^N \delta_{a_{t-1}}^{t-1}[j] Q^{t+1,j}$.
  $\mathsf{p}_{t+1} \leftarrow \textsc{StationaryDistribution}(Q^{t+1})$.

Then, the following holds:

$$\begin{aligned}
&\sum_{t=1}^T \mathop{\mathbb{E}}_{\mathsf{p}_t, dots, \mathsf{p}_1}[f_t] - \sum_{t=1}^T \min_{(x_t,\dots,x_1)} f_t(x_t,\dots,x_1) \\
&= \sum_{t=1}^T \mathop{\mathbb{E}}_{\mathsf{p}_t}[f_t] - \sum_{t=1}^T \min_{x_t} f_t(x_t) \\
&= \sum_{t=1}^T \mathop{\mathbb{E}}_{\mathsf{p}_t}[f_t] = \sum_{t=1}^T 1 - \min_j p_j \geq \sum_{t=1}^T 1 - 1/N = T(1-1/N) = \Omega(T),
\end{aligned}$$

which concludes the proof. □

B.2 Proof of Theorem 2.4

Proof. The cumulative loss can be written as follows in terms of $\mathcal{S}$:

$$\begin{aligned}\sum_{t=1}^{T} \mathsf{p}_t \cdot f_t &= \sum_{i,j=1}^{N} \sum_{t=1}^{T} \sum_{l=1}^{N} \mathsf{p}_t[i] \delta_{a_{t-1}}^{t-1}[j] Q_{i,l}^{t,j} f_t[l] \\ &= \sum_{t=1}^{T} \left(\sum_{(i,j)\in\mathcal{S}} + \sum_{(i,j)\notin\mathcal{S}} \right) \sum_{l=1}^{N} \mathsf{p}_t[i] \delta_{a_{t-1}}^{t-1}[j] Q_{i,l}^{t,j} f_t[l].\end{aligned}$$

Our algorithm, BigramStateDepCSwap, is the same as the one in the bigram case with the one caveat that all subroutines $A_{i,j}$ where $i, j \in \mathcal{S}$ must be derived from the same external regret minimizing algorithm. Then, as in the previous section, we can derive the bound

$$\begin{aligned}&\sum_{t=1}^{T} \sum_{(i,j)\notin\mathcal{S}} \sum_{l=1}^{N} \mathsf{p}_t[i] \delta_{a_{t-1}}^{t-1}[j] Q_{i,l}^{t,j} f_t[l] \\ &\quad \leq \sum_{(i,j)\notin\mathcal{S}} \left(\sum_{t=1}^{T} \mathsf{p}_t[i] \delta_{a_{t-1}}^{t-1}[j] f_t[\varphi(i,j)] + R_{i,j}(L_{\min}^{i,j}, N) \right).\end{aligned}$$

For the action pairs $(i, j) \in \mathcal{S}$, we can impose $Q_{i,l}^{t,j} = Q_{i,l}^{t,0}$ for all j, by associating to all algorithms the same loss $\frac{\sum_{j\colon (i,j)\in\mathcal{S}} \mathsf{p}_t[i] \delta_{a_{t-1}}^{t-1}[j] f_t}{|\{j\colon (i,j)\in\mathcal{S}\}|}$ and using the fact that all $\mathcal{A}_{i,j}$ are based off of the same subroutine.

With this choice of loss allocation, we can write

$$\begin{aligned}
&\sum_{t=1}^{T} \sum_{(i,j)\in\mathcal{S}} \sum_{l=1}^{N} \mathsf{p}_t[i]\delta_{a_{t-1}}^{t-1}[j]Q_{i,l}^{t,j} f_t[l] \\
&\qquad = \sum_{t=1}^{T} \sum_{i\colon \exists j,(i,j)\in\mathcal{S}} \sum_{l=1}^{N} \Big(\sum_{j\colon (i,j)\in\mathcal{S}} \mathsf{p}_t[i]\delta_{a_{t-1}}^{t-1}[j] f_t[l] \Big) Q_{i,l}^{t,0} \\
&\qquad \leq \sum_{i\colon \exists j,(i,j)\in\mathcal{S}} \left(\sum_{t=1}^{T} \Big(\sum_{j\colon (i,j)\in\mathcal{S}} \mathsf{p}_t[i]\delta_{a_{t-1}}^{t-1}[j] f_t[\tilde{\varphi}(i)] \Big) + R_i(L_{\min}^{i}, N) \right).
\end{aligned}$$

Combining the two terms yields

$$\begin{aligned}
\sum_{t=1}^{T} \mathsf{p}_t \cdot f_t \leq \sum_{t=1}^{T} \sum_{i,j=1}^{N} \mathsf{p}_t[i]\delta_{a_{t-1}}^{t-1}[j] f_t[\varphi(i,j)] + \sum_{(i,j)\notin\mathcal{S}} R_{i,j}(L_{\min}^{i,j}, N) \\
+ \sum_{i\colon \exists j,(i,j)\in\mathcal{S}} R_i(L_{\min}^{i}, N).
\end{aligned}$$

Next, using the fact that we allocated the original loss over the sub-algorithms, we can apply the standard bounds on external regret algorithms to obtain

$$\mathrm{Reg}_T(\mathcal{C}_{2,\mathcal{S}}, \textsc{BigramStateDepCSwap}) \leq \mathcal{O}(\sqrt{T(|\mathcal{S}^c| + N)\log(N)}).$$

□

B.3 Proof of Theorem 2.3

Proof. The result follows from a natural extension of the algorithm used in Theorem 2.2. The pseudocode is provided as KgramCSwap.

Using this distribution and proceeding otherwise as in the proof of Theorem 2.2 to bound the cumulative loss leads to the desired inequality.

Algorithm 21: KGRAMCSWAP.

Algorithm: KGRAMCSWAP($(\mathcal{A}_{j_0,\dots,j_{k-1}})_{j_0,\dots,j_{k-1}=1}^{N}$), $(\mathcal{A}_{j_0,\dots,j_{k-1}})_{j_0,\dots,j_{k-1}=1}^{N}$ external-regret minimizing online learning algorithms.

$\mathsf{p}_1 \leftarrow \frac{1}{N}$.
$a_0 \leftarrow \textsc{Sample}(\mathsf{p}_1)$.
for $t \leftarrow 1, \dots, T$ **do**
  $a_t \leftarrow \textsc{Sample}(\mathsf{p}_t)$.
  $f_t \leftarrow \textsc{Play}(a_t)$.
  for $j_0, \dots, j_{k-1} \leftarrow 1, \dots, N$ **do**
    $\textsc{AssignLoss}(\mathsf{p}_t[j_0]\delta_{a_{t-1}}^{t-1}[j_1] \dots \delta_{a_{t-(k-1)}}^{t-(k-1)}[j_{k-1}]f_t, \mathcal{A}_{j_0,\dots,j_{k-1}})$.
    $Q_{j_0}^{t+1,j_1,\dots,j_{k-1}} \leftarrow \textsc{AlgoStrategy}(\mathcal{A}_{j_0,\dots,j_{k-1}})$.
  $Q^{t+1} \leftarrow \sum_{j_1,\dots,j_{k-1}=1}^{N} \delta_{a_{t-1}}^{t-1}[j_1] \dots \delta_{a_{t-(k-1)}}^{t-(k-1)}[j_{k-1}]Q^{t+1,j_1,\dots,j_{k-1}}$.
  $\mathsf{p}_{t+1} \leftarrow \textsc{StationaryDistribution}(Q^{t+1})$.

□

B.4 Proof of Theorem 2.6

Proof. Let

$$D_\epsilon = \{(s,r) \in \Sigma_k^2 \mid \exists \varphi_k \colon \Sigma_k^2 \to \Sigma_k \text{ s.t. } l^{(k)}(s_k, s_{(-k)}) - l^{(k)}(\varphi_k(s_k, r_k), s_{(-k)}) \geq \epsilon\}$$

denote the set of action pairs that are conditionally ϵ-dominated. Then $\widehat{P}^T(D_\epsilon) = \frac{1}{T}\sum_{t=1}^{T}\sum_{(s,r)\in D_\epsilon} \delta_{s_k^t}[s]\delta_{s_k^{t-1}}[r]$ is the total empirical mass of D_ϵ.

Thus, we can write

$$\begin{aligned}
\epsilon T \widehat{P}^T(D_\epsilon) &= \epsilon \sum_{t=1}^{T} \sum_{(s,r)\in D_\epsilon} \delta_{s_k^t}[s]\delta_{s_k^{t-1}}[r] \\
&\leq \sum_{t=1}^{T} \sum_{(s,r)\in D_\epsilon} \mathbb{E}[l^{(k)}(s_k^t, s_{(-k)}^t) - l^{(k)}(\varphi_k(s_k^t, s_k^{t-1}), s_{(-k)}^t)] \\
&\leq \max_{\varphi} \sum_{t=1}^{T} \sum_{(s,r)\in D_\epsilon} \mathbb{E}[l^{(k)}(s_k^t, s_{(-k)}^t) - l^{(k)}(\varphi(s_k^t, s_k^{t-1}), s_{(-k)}^t)] \\
&= \mathrm{reg}(k, T),
\end{aligned}$$

and the conditional swap regret of player k, $\mathrm{reg}(k, T)$, satisfies

$$\mathrm{reg}(k, T) \geq \epsilon T \widehat{P}^T(D_\epsilon).$$

Furthermore, since player k is following a conditional swap regret minimizing strategy, we must have $\mathrm{reg}(k, T) \leq R$. This implies that $R \geq \epsilon T \widehat{P}^T(D_\epsilon)$. □

B.5 Proof of Theorem 2.7

Proof. As in the full information scenario, we can design a meta-algorithm that combines several external regret minimizing subroutines. The difference in the bandit scenario is that we must use a surrogate loss since we do not have access to the loss of every action at each round. Our algorithm is presented as BanditBigramCSwap, where the surrogate loss is denoted as $\widehat{f}_t$.

Algorithm 22: BanditBigramCSwap.

Algorithm: BanditBigramCSwap$((\mathcal{A}_{i,j})_{i,j=1}^N)$,
$(\mathcal{A}_{i,j})_{i,j=1}^N$ external-regret minimizing online learning algorithms.
$\mathsf{p}_1 \leftarrow \frac{1}{N}$.
$a_0 \leftarrow \textsc{Sample}(\mathsf{p}_1)$.
for $t \leftarrow 1, \dots, T$ **do**
  $a_t \leftarrow \textsc{Sample}(\mathsf{p}_t)$.
  $f_t(a_t) \leftarrow \textsc{Play}(a_t)$.
  $\widehat{f_t} \leftarrow \frac{f_t(a_t) 1_{a_t}}{\mathsf{p}_t}$.
  for $i, j \leftarrow 1, \dots, N$ **do**
    $\textsc{AssignLoss}(\mathsf{p}_t[i]\delta_{a_{t-1}}^{t-1}[j]\widehat{f_t}, \mathcal{A}_{i,j})$.
    $Q_i^{t+1,j} \leftarrow \textsc{AlgoStrategy}(\mathcal{A}_{i,j})$.
  $Q^{t+1} \leftarrow \sum_{j=1}^N \delta_{a_{t-1}}^{t-1}[j] Q^{t+1,j}$.
  $\mathsf{p}_{t+1} \leftarrow \textsc{StationaryDistribution}(Q^{t+1})$.

This algorithm allows us to compute

$$\begin{aligned}
\sum_{t=1}^T \mathop{\mathbb{E}}_{x_t \sim \mathsf{p}_t}[f_t(x_t)] &= \sum_{t=1}^T \sum_{l=1}^N \mathsf{p}_t[l] f_t[l] \\
&= \sum_{t=1}^T \sum_{i,j,l=1}^N \mathsf{p}_t[i] \delta_{a_{t-1}}^{t-1}[j] Q_{i,l}^{t,j} f_t[l] \\
&= \sum_{i,j=1}^N \sum_{t=1}^T \sum_{l=1}^N \left(\mathsf{p}_t[i] \delta_{a_{t-1}}^{t-1}[j] f_t[l]\right) Q_{i,l}^{t,j} \\
&= \sum_{i,j=1}^N \sum_{t=1}^T \mathop{\mathbb{E}}_{x_t \sim Q_i^{t,j}} \left[\mathsf{p}_t[i] \delta_{a_{t-1}}^{t-1}[j] \widehat{f_t}(x_t)\right] \\
&\leq \sum_{i,j=1}^N \sum_{t=1}^T \left[\left(\sum_{t=1}^T \mathsf{p}_t[i] \delta_{a_{t-1}}^{t-1}[j] f_t[\varphi(i,j)]\right) + 2\sqrt{L_{\min}^{i,j} N \log(N)}\right] \\
&= \sum_{t=1}^T \mathop{\mathbb{E}}_{(x_t, x_{t-1}) \sim (\mathsf{p}_t, \delta_{a_{t-1}}^{t-1})} [f_t(\varphi(x_t, x_{t-1}))] + 2N^2 \sqrt{L_{\min}^{i,j} N \log(N)},
\end{aligned}$$

where the inequality comes from estimate (3.3) provided by Theorem 3.1 in Bubeck et al. [2012]. By applying the same convexity argument as in Theorem 2.2, we can

Algorithm 23: BANDITKGRAMCSWAP.

Algorithm: BANDITKGRAMCSWAP($(\mathcal{A}_{j_0,\dots,j_{k-1}})_{j_0,\dots,j_{k-1}=1}^{N}$), $(\mathcal{A}_{j_0,\dots,j_{k-1}})_{j_0,\dots,j_{k-1}=1}^{N}$ external-regret minimizing online learning algorithms.

$\mathsf{p}_1 \leftarrow \frac{1}{N}$.
$a_0 \leftarrow \textsc{Sample}(\mathsf{p}_1)$.
for $t \leftarrow 1, \dots, T$ **do**
  $a_t \leftarrow \textsc{Sample}(\mathsf{p}_t)$.
  $f_t(a_t) \leftarrow \textsc{Play}(a_t)$.
  $\widehat{f}_t \leftarrow \frac{f_t(a_t)1_{a_t}}{\mathsf{p}_t}$.
  for $j_0, \dots, j_{k-1} \leftarrow 1, \dots, N$ **do**
    $\textsc{AssignLoss}(\mathsf{p}_t[j_0]\delta_{a_{t-1}}^{t-1}[j_1]\dots\delta_{a_{t-(k-1)}}^{t-(k-1)}[j_{k-1}]\widehat{f}_t, \mathcal{A}_{j_0,\dots,j_{k-1}})$.
    $Q_{j_0}^{t+1,j_1,\dots,j_{k-1}} \leftarrow \textsc{AlgoStrategy}(\mathcal{A}_{j_0,\dots,j_{k-1}})$.
  $Q^{t+1} \leftarrow \sum_{j_1,\dots,j_{k-1}=1}^{N} \delta_{a_{t-1}}^{t-1}[j_1]\dots\delta_{a_{t-(k-1)}}^{t-(k-1)}[j_{k-1}]Q^{t+1,j_1,\dots,j_{k-1}}$.
  $\mathsf{p}_{t+1} \leftarrow \textsc{StationaryDistribution}(Q^{t+1})$.

refine the bound to get

$$\sum_{t=1}^{T} \mathop{\mathbb{E}}_{\mathsf{p}_t}[f_t] - \sum_{t=1}^{T} \mathop{\mathbb{E}}_{(x_t,x_{t-1})\sim(\mathsf{p}_t,\delta_{a_{t-1}}^{t-1})}[f_t(\varphi(x_t,x_{t-1}))] \leq 2\sqrt{N^2 L_{\min} N \log(N)}$$
$$\leq 2\sqrt{N^3 T \log(N)}$$

as desired.

□

B.6 Proof of Theorem 2.8

Proof. As in the full information case, the result follows from a natural extension of the algorithm used in Theorem 2.7 and is analogous to the algorithm used in Theorem 2.3. The algorithm is presented as BANDITKGRAMCSWAP.

Using this distribution and proceeding otherwise as in the proof of Theorem 2.7 to bound the cumulative loss leads to the desired inequality. □

Appendix C

Supplementary material for Chapter 3

C.1 Further discussion and proofs for Section 3.2

C.1.1 Technical lemma on convex duality

Lemma C.1 (Duality Between Smoothness and Convexity for Convex Functions)**.** *Let $\mathcal{K}$ be a convex set and $f : \mathcal{K} \to \mathbb{R}$ be a convex function. Suppose f is* 1*-strongly convex at x_0. Then f^*, the Legendre transform of f, is* 1*-strongly smooth at $y_0 = \nabla f(x_0)$.*

Proof. Notice first that for any pair of convex functions $f, g : \mathcal{K} \to \mathbb{R}$, the fact that $f(x_0) \geq g(x_0)$ for some $x_0 \in \mathcal{K}$ implies that $f^*(y_0) \leq g^*(y_0)$ for $y_0 = \nabla f(x_0)$.

Now, f being 1-strongly convex at x_0 means that $f(x) \geq h(x) = f(x_0) + g_0^\top (x - x_0) + \frac{\sigma}{2}\|x - x_0\|_2^2$. Thus, it suffices to show that $h^*(y) = f^*(y_0) + x_0^\top (y - y_0) + \frac{1}{2}\|y - y_0\|_2^2$, since $x_0 = \nabla(h^*)(y_0)$.

To see this, we can compute

$$\begin{aligned}
h^*(y) &= \max_x y^\top x - h(x) \\
&= y^\top (y - y_0 + x_0) - h(x) \\
&\quad \text{(max attained at } y_0 + (x - x_0) = \nabla h(x) = y) \\
&= y^\top (y - y_0 + x_0) \\
&\quad - \left[f(x_0) + y_0^\top (x - x_0) + \frac{1}{2}\|x - x_0\|_2^2 \right] \\
&= \frac{1}{2}\|y - y_0\|_2^2 + y^\top x_0 - f(x_0) \\
&= -f(x_0) + x_0^\top y_0 + x_0^\top (y - y_0) + \frac{1}{2}\|y - y_0\|_2^2 \\
&= f^*(y_0) + x_0^\top (y - y_0) + \frac{1}{2}\|y - y_0\|_2^2.
\end{aligned}$$

□

C.1.2 Proof of Theorem 3.2 for AdaOptFTRL with general regularization

Proof. Recall that $x_{t+1} = \operatorname{argmin}_{x \in \mathcal{K}} x^\top (g_{1:t} + \tilde{g}_{t+1}) + r_{0:t}(x)$, and let $y_t = \operatorname{argmin}_{x \in \mathcal{K}} x^\top g_{1:t} + r_{0:t-1}(x)$. Then by convexity,

$$\begin{aligned}
\sum_{t=1}^T f_t(x_t) - f_t(x) &\le \sum_{t=1}^T g_t^\top (x_t - x) \\
&= \sum_{t=1}^T (g_t - \tilde{g}_t)^\top (x_t - y_t) + \tilde{g}_t^\top (x_t - y_t) + g_t^\top (y_t - x).
\end{aligned}$$

Now, we first show via induction that $\forall x \in \mathcal{K}$, the following holds:

$$\sum_{t=1}^{T} \tilde{g}_t^\top (x_t - y_t) + g_t^\top y_t \leq \sum_{t=1}^{T} g_t^\top x + r_{0:T-1}(x).$$

For $T = 1$, the fact that $r_t \geq 0$, $\tilde{g}_1 = 0$, and the definition of y_t imply the result.

Now suppose the result is true for time T. Then

$$\begin{aligned}
&\sum_{t=1}^{T+1} \tilde{g}_t^\top (x_t - y_t) + g_t^\top y_t \\
&= \left[\sum_{t=1}^{T} \tilde{g}_t^\top (x_t - y_t) + g_t^\top y_t\right] + \tilde{g}_{T+1}^\top (x_{T+1} - y_{T+1}) + g_{T+1}^\top y_{T+1} \\
&\leq \left[\sum_{t=1}^{T} g_t^\top x_{T+1} + r_{0:T-1}(x_{T+1})\right] + \tilde{g}_{T+1}^\top (x_{T+1} - y_{T+1}) + g_{T+1}^\top y_{T+1} \\
&\quad \text{(by the induction hypothesis for } x = x_{T+1}) \\
&\leq \left[(g_{1:T} + \tilde{g}_{T+1})^\top x_{T+1} + r_{0:T}(x_{T+1})\right] + \tilde{g}_{T+1}^\top (-y_{T+1}) + g_{T+1}^\top y_{T+1} \\
&\quad \text{(since } r_t \geq 0, \forall t) \\
&\leq \left[(g_{1:T} + \tilde{g}_{T+1})^\top y_{T+1} + r_{0:T}(y_{T+1})\right] + \tilde{g}_{T+1}^\top (-y_{T+1}) + g_{T+1}^\top y_{T+1} \\
&\quad \text{(by definition of } x_{T+1}) \\
&\leq g_{1:T+1}^\top y + r_{0:T}(y), \text{ for any } y \\
&\quad \text{(by definition of } y_{T+1}).
\end{aligned}$$

Thus, we have $\sum_{t=1}^{T} f_t(x_t) - f_t(x) \leq r_{0:T-1}(x) + \sum_{t=1}^{T} (g_t - \tilde{g}_t)^\top (x_t - y_t)$ and it suffices to bound $\sum_{t=1}^{T} (g_t - \tilde{g}_t)^\top (x_t - y_t)$. By duality again, one can immediately get $(g_t - \tilde{g}_t)^\top (x_t - y_t) \leq \|g_t - \tilde{g}_t\|_{(t-1),*} \|x_t - y_t\|_{(t-1)}$. To bound $\|x_t - y_t\|_{(t)}$ in terms

of the gradient, recall first that

$$x_t = \operatorname*{argmin}_{x \in \mathcal{K}} h_{0:t-1}(x)$$
$$y_t = \operatorname*{argmin}_{x \in \mathcal{K}} h_{0:t-1}(x) + (g_t - \widehat{g}_t)^\top x.$$

The fact that $r_{0:t-1}(x)$ is 1-strongly convex with respect to the norm $\|\cdot\|_{(t-1)}$ implies that $h_{0:t-1}$ is as well. In particular, it is strongly convex at the points x_t and y_t. But, this then implies that the conjugate function is smooth at $\nabla(h_{0:t-1})(x_t)$ and $\nabla(h_{0:t-1})(y_t)$, so that

$$\|\nabla(h^*_{0:t-1})(-(g_t - \tilde{g}_t)) - \nabla(h^*_{0:t-1})(0)\|_{(t)} \leq \|g_t - \tilde{g}_t\|_{(t-1),*}$$

Since $\nabla(h^*_{0:t-1})(-(g_t - \tilde{g}_t)) = y_t$ and $\nabla(h^*_{0:t-1})(0) = x_t$, we have $\|x_t - y_t\|_{(t-1)} \leq \|g_t - \tilde{g}_t\|_{(t-1),*}$.

□

C.1.3 Proof of Theorem 3.3 for CoAdaOptFTRL with proximal regularization

Proof. For the first regret bound, define the auxiliary regularization functions $\tilde{r}_t(x) = r_t(x) + \psi_t(x)$, and apply Theorem 3.2 to get

$$\begin{aligned}\sum_{t=1}^{T} f_t(x_t) - f_t(x) &\leq \tilde{r}_{0:T-1}(x) + \sum_{t=1}^{T} \|g_t - \tilde{g}_t\|^2_{(t-1),*} \\ &= \psi_{1:T-1}(x) + r_{0:T-1}(x) + \sum_{t=1}^{T} \|g_t - \tilde{g}_t\|^2_{(t-1),*}\end{aligned}$$

Notice that while r_t is proximal, $\tilde{r}_t$, in general, is not, and so we must apply the theorem with general regularizers instead of the one with proximal regularizers.

For the second regret bound, we can follow the prescription of Theorem 1 while keeping track of the additional composite terms:

Recall that $x_{t+1} = \operatorname{argmin}_{x \in \mathcal{K}} x^\top (g_{1:t} + \tilde{g}_{t+1}) + r_{0:t+1}(x) + \psi_{1:t+1}(x)$, and let $y_t = \operatorname{argmin}_{x \in \mathcal{K}} x^\top g_{1:t} + r_{0:t}(x) + \psi_{1:t}(x)$.

We can compute:

$$\begin{aligned}\sum_{t=1}^{T} f_t(x_t) + \psi(x_t) - [f_t(x) + \psi_t(x)] &\leq \sum_{t=1}^{T} g_t^\top (x_t - x) + \psi_t(x_t) - \psi_t(x) \\ &= \sum_{t=1}^{T} (g_t - \tilde{g}_t)^\top (x_t - y_t) \\ &\quad + \tilde{g}_t^\top (x_t - y_t) + g_t^\top (y_t - x) + \psi_t(x_t) - \psi_t(x)\end{aligned}$$

Similar to before, we show via induction that $\forall x \in \mathcal{K}$, $\sum_{t=1}^{T} \tilde{g}_t^\top (x_t - y_t) + g_t^\top y_t + \psi_t(x_t) \leq r_{0:T}(x) + \sum_{t=1}^{T} g_t^\top x + \psi_t(x)$.

For $T = 1$, the fact that $r_t \geq 0$, $\widehat{g}_1 = 0$, $\psi_1(x_1) = 0$, and the definition of y_t imply the result.

Now suppose the result is true for time T. Then

$$
\begin{aligned}
&\sum_{t=1}^{T+1} \tilde{g}_t^\top (x_t - y_t) + g_t^\top y_t + \psi_t(x_t) \\
&= \left[\sum_{t=1}^{T} \tilde{g}_t^\top (x_t - y_t) + g_t^\top y_t + \psi_t(x_t) \right] + \tilde{g}_{T+1}^\top (x_{T+1} - y_{T+1}) + g_{T+1}^\top y_{T+1} \\
&\quad + \psi_{T+1}(x_{T+1}) \\
&\leq \left[\sum_{t=1}^{T} g_t^\top x_{T+1} + r_{0:T}(x_{T+1}) + \psi_t(x_{T+1}) \right] + \tilde{g}_{T+1}^\top (x_{T+1} - y_{T+1}) + g_{T+1}^\top y_{T+1} \\
&\quad + \psi_{T+1}(x_{T+1}) \\
&\quad \text{(by the induction hypothesis for } x = x_{T+1}) \\
&\leq (g_{1:T} + \tilde{g}_{T+1})^\top x_{T+1} + r_{0:T+1}(x_{T+1}) + \psi_t(x_{T+1}) + \tilde{g}_{T+1}^\top (-y_{T+1}) + g_{T+1}^\top y_{T+1} \\
&\quad + \psi_{T+1}(x_{T+1}) \\
&\quad \text{(since } r_t \geq 0, \forall t) \\
&\leq (g_{1:T} + \tilde{g}_{T+1})^\top y_{T+1} + r_{0:T+1}(y_{T+1}) + \psi_t(y_{T+1}) + \tilde{g}_{T+1}^\top (-y_{T+1}) + g_{T+1}^\top y_{T+1} \\
&\quad + \psi_{T+1}(y_{T+1}) \\
&\quad \text{(by definition of } x_{T+1}) \\
&\leq g_{1:T+1}^\top y + r_{0:T+1}(y) + \psi_{1:T+1}(y), \text{ for any } y \\
&\quad \text{(by definition of } y_{T+1}).
\end{aligned}
$$

Thus, we have

$$\sum_{t=1}^{T} f_t(x_t) + \psi_t(x_t) - [f_t(x) + \psi_t(x)] \leq r_{0:T}(x) + \sum_{t=1}^{T} (g_t - \tilde{g}_t)^\top (x_t - y_t),$$

and we can bound the sum in the same way as before, since the strong convexity

properties of $h_{0:t}$ are retained due to the convexity of ψ_t.

□

C.1.4 CoAdaOptFTRL for general regularization

Theorem C.1 (CoAdaOptFTRL guarantee for general regularization)**.** *Let $\{r_t\}_{t=1}^T$ be a sequence of non-negative functions, and let $\tilde{g}_t$ be the learner's estimate of g_t given the history of functions $f_1, \dots, f_{t-1}$ and points $x_1, \dots, x_{t-1}$. Let $\{\psi_t\}_{t=1}^T$ be a sequence of non-negative convex functions such that $\psi_1(x_1) = 0$. Assume further that the function $h_{0:t} : x \mapsto g_{1:t}^\top x + \tilde{g}_{t+1}^\top x + r_{0:t}(x) + \psi_{1:t+1}(x)$ is 1-strongly convex with respect to some norm $\|\cdot\|_{(t)}$. Then, the following regret bound holds for* CoAdaOptFTRL *(Algorithm 16):*

$$\sum_{t=1}^T f_t(x_t) - f_t(x) \le \psi_{1:T-1}(x) + r_{0:T-1}(x) + \sum_{t=1}^T \|g_t - \tilde{g}_t\|_{(t-1),*}^2$$
$$\sum_{t=1}^T f_t(x_t) + \psi_t(x_t) - [f_t(x) + \psi_t(x)] \le r_{0:T-1}(x) + \sum_{t=1}^T \|g_t - \tilde{g}_t\|_{(t),*}^2 .$$

Proof. For the first regret bound, define the auxiliary regularization functions $\tilde{r}_t(x) = r_t(x) + \psi(x)$, and apply Theorem 3.2 to get

$$\begin{aligned}\sum_{t=1}^T f_t(x_t) - f_t(x) &\le \tilde{r}_{0:T-1}(x) + \sum_{t=1}^T \|g_t - \widehat{g}_t\|_{(t),*}^2 \\ &= \psi_{1:T-1}(x) + r_{0:T-1}(x) + \sum_{t=1}^T \|g_t - \widehat{g}_t\|_{(t-1),*}^2.\end{aligned}$$

For the second bound, we can proceed as in the original proof, but now keep track of the additional composite terms.

Recall that $x_{t+1} = \operatorname{argmin}_{x \in \mathcal{K}} x^\top (g_{1:t} + \tilde{g}_{t+1}) + r_{0:t}(x) + \psi_{1:t+1}(x)$, and let $y_t = \operatorname{argmin}_{x \in \mathcal{K}} x^\top g_{1:t} + r_{0:t-1}(x) + \psi_{1:t}(x)$. Then

$$\begin{aligned}\sum_{t=1}^{T} f_t(x_t) + \psi_t(x_t) - f_t(x) - \psi_t(x) &\le \sum_{t=1}^{T} g_t^\top (x_t - x) + \psi_t(x_t) - \psi_t(x) \\ &= \sum_{t=1}^{T} (g_t - \tilde{g}_t)^\top (x_t - y_t) + \tilde{g}_t^\top (x_t - y_t) \\ &\quad + g_t^\top (y_t - x) + \psi_t(x_t) - \psi_t(x).\end{aligned}$$

Now, we show via induction that $\forall x \in \mathcal{K}$, $\sum_{t=1}^{T} \tilde{g}_t^\top (x_t - y_t) + g_t^\top y_t + \psi(x_t) \le \sum_{t=1}^{T} g_t^\top x + \psi_t(x) + r_{0:T-1}(x)$.

For $T = 1$, the fact that $r_t \ge 0$, $\widehat{g}_1 = 0$, $\psi_1(x_1) = 0$, and the definition of y_t imply the result.

Now suppose the result is true for time T. Then

$$
\begin{aligned}
&\sum_{t=1}^{T+1} \tilde{g}_t^\top (x_t - y_t) + g_t^\top y_t + \psi_t(x_t) \\
&= \left[\sum_{t=1}^{T} \tilde{g}_t^\top (x_t - y_t) + g_t^\top y_t + \psi_t(x_t) \right] + \tilde{g}_{T+1}^\top (x_{T+1} - y_{T+1}) + g_{T+1}^\top y_{T+1} \\
&\quad + \psi_{T+1}(x_{T+1}) \\
&\leq \left[\sum_{t=1}^{T} g_t^\top x_{T+1} + r_{0:T-1}(x_{T+1}) + \psi_t(x_{T+1}) \right] + \tilde{g}_{T+1}^\top (x_{T+1} - y_{T+1}) + g_{T+1}^\top y_{T+1} \\
&\quad + \psi_{T+1}(x_{T+1}) \\
&\quad \text{(by the induction hypothesis for } x = x_{T+1}) \\
&\leq \left[(g_{1:T} + \tilde{g}_{T+1})^\top x_{T+1} + r_{0:T}(x_{T+1}) + \psi_t(x_{T+1}) \right] + \tilde{g}_{T+1}^\top (-y_{T+1}) + g_{T+1}^\top y_{T+1} \\
&\quad + \psi_{T+1}(x_{T+1}) \\
&\text{(since } r_t \geq 0, \, \forall t) \\
&\leq g_{1:T+1}^\top y_{T+1} + \tilde{g}_{T+1}^\top y_{T+1} + r_{0:T}(y_{T+1}) + \tilde{g}_{T+1}^\top (-y_{T+1}) + g_{T+1}^\top y_{T+1} \\
&\quad + \psi_{1:T+1}(y_{T+1}) \\
&\quad \text{(by definition of } x_{T+1}) \\
&\leq g_{1:T+1}^\top y + r_{0:T}(y) + \psi_{1:T+1}(y), \text{ for any } y \\
&\quad \text{(by definition of } y_{T+1}).
\end{aligned}
$$

Thus, we have $\sum_{t=1}^{T} f_t(x_t) + \psi_t(x_t) - f_t(x) - \psi_t(x) \leq r_{0:T-1}(x) + \sum_{t=1}^{T} (g_t - \tilde{g}_t)^\top (x_t - y_t)$ and the remainder follows as in the non-composite setting since the strong convexity properties are retained.

□

C.1.5 Technical lemma for Section 3.2.2.1 on AdaOptOGD

The following lemma is central to the derivation of regret bounds for many algorithms employing adaptive regularization (e.g. [Duchi et al., 2010, McMahan and Streeter, 2010]).

Lemma C.2. *Let $\{a_j\}_{j=1}^T$ be a sequence of non-negative numbers. Then*

$$\sum_{j=1}^{T} \frac{a_j}{\sqrt{\sum_{k=1}^{j} a_k}} \leq 2\sqrt{\sum_{j=1}^{T} a_j}.$$

Proof. We prove this result using induction. For $T = 1$, $\sqrt{a_1} \leq 2\sqrt{a_1}$ is always true. Now assume the result for $T - 1$. Then

$$\begin{aligned}\sum_{j=1}^{T} \frac{a_j}{\sqrt{\sum_{k=1}^{j} a_k}} &= \left(\sum_{j=1}^{T-1} \frac{a_j}{\sqrt{\sum_{k=1}^{j} a_k}}\right) + \frac{a_T}{\sqrt{\sum_{k=1}^{T} a_k}} \\ &\leq 2\sqrt{\sum_{j=1}^{T-1} a_j} + \frac{a_T}{\sqrt{\sum_{k=1}^{T} a_k}}.\end{aligned}$$

Now let, $Z = \sum_{j=1}^{T} a_j$ and $x = a_T$. Then we can rewrite this last expression as:

$$2\sqrt{Z - x} + \frac{x}{\sqrt{Z}}.$$

This last expression is concave in x and maximized when $x = 0$. Thus, it follows that:

$$2\sqrt{\sum_{j=1}^{T-1} a_j} + \frac{a_T}{\sqrt{\sum_{k=1}^{T} a_k}} \leq 2\sqrt{\sum_{j=1}^{T} a_j}.$$

□

C.2 Further discussion and proofs for Section 3.3

C.2.1 Proof of Theorem 3.4 for CoAdaOptFTRL with stochastic subgradients and proximal regularization

Proof.

$$\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^{T} f_t(x_t) - f_t(x)\right] &\leq \sum_{t=1}^{T} \mathbb{E}\left[g_t^\top (x_t - x)\right] \\
&= \sum_{t=1}^{T} \mathbb{E}\left[\mathbb{E}[\widehat{g}_t | \widehat{g}_1, \ldots, \widehat{g}_{t-1}, x_1, \ldots, x_t]^\top (x_t - x)\right] \\
&= \sum_{t=1}^{T} \mathbb{E}\left[\mathbb{E}[\widehat{g}_t^\top (x_t - x) | \widehat{g}_1, \ldots, \widehat{g}_{t-1}, x_1, \ldots, x_t]\right] \\
&= \sum_{t=1}^{T} \mathbb{E}\left[\widehat{g}_t^\top (x_t - x)\right].
\end{aligned}$$

This implies that upon taking an expectation, we can freely upper bound the difference $f_t(x_t) - f_t(x)$ by the noisy linearized estimate $\widehat{g}_t^\top (x_t - x)$. After that, we can apply Algorithm 16 on the gradient estimates to get the bounds:

$$\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^{T} \widehat{g}_t^\top (x_t - x)\right] &\leq \mathbb{E}\left[\psi_{1:T-1}(x) + r_{0:T-1}(x) + \sum_{t=1}^{T} \|\widehat{g}_t - \tilde{g}_t\|_{(t-1),*}^2\right] \\
\mathbb{E}\left[\sum_{t=1}^{T} \widehat{g}_t^\top (x_t - x) + \psi_t(x_t) - \psi_t(x)\right] &\leq \mathbb{E}\left[r_{0:T}(x) + \sum_{t=1}^{T} \|\widehat{g}_t - \tilde{g}_t\|_{(t),*}^2\right].
\end{aligned}$$

□

C.2.2 CoAdaOptFTRL with stochastic subgradients and general regularization

Theorem C.2 (CoAdaOptFTRL stochastic general guarantee)**.** *Let* $\{r_t\}_{t=1}^T$ *be a sequence of non-negative functions, and let* $\tilde{g}_t$ *be the learner's estimate of* $\widehat{g}_t$ *given the history of noisy gradients* $\widehat{g}_1, \dots, \widehat{g}_{t-1}$ *and points* $x_1, \dots, x_{t-1}$*. Let* $\{\psi_t\}_{t=1}^T$ *be a sequence of non-negative convex functions, such that* $\psi_1(x_1) = 0$*. Assume furthermore that the function* $h_{0:t}(x) = \widehat{g}_{1:t}^\top x + \tilde{g}_{t+1}^\top x + r_{0:t}(x) + \psi_{1:t+1}(x)$ *is 1-strongly convex with respect to some norm* $\|\cdot\|_{(t)}$*. Then, the update* $x_{t+1} = \operatorname{argmin}_{x \in \mathcal{K}} h_{0:t}(x)$ *of* CoAdaOptFTRL *with stochastic subgradients yields the regret bounds:*

$$\mathbb{E}\left[\sum_{t=1}^T f_t(x_t) - f_t(x)\right] \le \mathbb{E}\left[\psi_{1:T-1}(x) + r_{0:T-1}(x) + \sum_{t=1}^T \|\widehat{g}_t - \tilde{g}_t\|_{(t-1),*}^2\right]$$
$$\mathbb{E}\left[\sum_{t=1}^T f_t(x_t) + \psi_t(x_t) - f_t(x) - \psi_t(x)\right] \le \mathbb{E}\left[r_{0:T-1}(x) + \sum_{t=1}^T \|\widehat{g}_t - \tilde{g}_t\|_{(t-1),*}^2\right].$$

Proof. The argument is the same as for Theorem 3.4, except that we now apply the bound of Theorem C.1 at the end. □

Algorithm 24: AdaOptRegERM-Batch.

Algorithm: AdaOptRegERM-Batch(λ, r_0, ψ, $(p_t)_{t=1}^T$, $\{\Pi_j\}_{j=1}^l$, K, x_1), λ scaling constant, r_0 regularization function, ψ composite function, $(p_t)_{t=1}^T$ distributions over $\{1,\ldots,m\}$, $\{\Pi_j\}_{j=1}^l$ partition of $\{1,\ldots,m\}$, K number of epochs, $x_1 \in \mathcal{K}$ initial point.

$j_1 \leftarrow \text{Sample}(p_1)$. $t \leftarrow 1$. **for** $s \leftarrow 1,\ldots,k$ **do**
  for $j \leftarrow 1,\ldots,m$ **do**
    $\bar{g}_s^j \leftarrow \nabla f_j(x_1)$.
  for $a \leftarrow 1,\ldots,\frac{T}{k}$ **do**
    if $T \mod k = 0$ **then**
      for $j \leftarrow 1,\ldots,m$ **do**
        $g^j \leftarrow \nabla f_j(x_t)$.
    $\widehat{g}_t \leftarrow \frac{\sum_{j\in\Pi_{j_t}} g_t^j}{p_{t,j_t}}$.
    $r_t \leftarrow \text{DesignRegularizer}((\widehat{g}_s)_{s=1}^t, (x_s)_{s=1}^t)$.
    $j_{t+1} \leftarrow \text{Sample}(p_{t+1})$.
    $\tilde{g}_{t+1} \leftarrow \frac{\sum_{j\in\Pi_{j_t}} \bar{g}_s^j}{p_{t,j_t}}$.
    $x_{t+1} \leftarrow \text{argmin}_{x\in\mathcal{K}} \widehat{g}_{1:t}^\top x + \tilde{g}_{t+1}^\top x + r_{0:t}(x) + (t+1)\lambda\psi(x)$.
    $t \leftarrow t+1$.

C.3 Further discussion for Section 3.3.2.2

C.3.1 AdaOptRegERM-Batch, a mini-batch version of AdaOptRegERM

Corollary C.1. *Assume* $\mathcal{K} \subset \times_{i=1}^n [-R_i, R_i]$. *Let* $\cup_{j=1}^l \{\Pi_j\} = \{1,\ldots,n\}$ *be a partition of the functions* f_i, *and let* $e_{\Pi_j} = \sum_{i\in\Pi_j} e_i$. *Denote*

$$\Delta_{s,i} = \sqrt{\sum_{a=1}^{s} (\widehat{g}_{a,i} - \tilde{g}_{a,i})^2},$$

and let

$$r_{0:t} = \sum_{i=1}^{n} \sum_{s=1}^{t} \frac{\Delta_{s,i} - \Delta_{s-1,i}}{2R_i} (x_i - x_{s,i})^2$$

be the adaptive regularization.

Then the regret of Algorithm 24 is bounded by:

$$\mathbb{E}\left[\sum_{t=1}^{T} f_t(x_t) + \lambda\psi(x_t) - f_t(x) - \lambda\psi(x)\right]$$
$$\leq \sum_{i=1}^{n} 4R_i \sqrt{\sum_{s=1}^{K} \sum_{t=\frac{(s-1)T}{K}+1}^{\frac{sT}{K}} \sum_{a=1}^{l} \frac{\left|\sum_{j\in\Pi_j} g_{t,i}^{j} - \bar{g}_{s,i}^{j}\right|^2}{p_{t,a}}}.$$

Moreover, if $\|\nabla f_j\|_\infty \leq L_j \ \forall j$*, then setting* $p_{t,j} = \frac{L_i}{\sum_{j=1}^{m} L_j}$ *yields a worst-case bound of:* $8\sum_{i=1}^{n} R_i \sqrt{T\left(\sum_{j=1}^{m} L_j\right)^2}$.

A similar approach to Regularized ERM was developed independently by [Zhao and Zhang, 2015]. However, the one here improves upon that algorithm through the incorporation of adaptive regularization, optimistic gradient predictions, and the fact that we do not assume higher regularity conditions such as strong convexity for our loss functions.

Bibliography

Dmitry Adamskiy, Wouter M Koolen, Alexey Chernov, and Vladimir Vovk. A closer look at adaptive regret. In *International Conference on Algorithmic Learning Theory*, pages 290–304. Springer, 2012.

Arash Afkanpour, András György, Csaba Szepesvári, and Michael Bowling. A randomized mirror descent algorithm for large scale multiple kernel learning. In *ICML*, JMLR Proceedings, pages 374–382, 2013.

A. V. Aho and M. J. Corasick. Efficient string matching: An aid to bibliographic search. *Communication of the Association for Computing Machinery*, 18 (6): 333–340, 1975.

Cyril Allauzen and Mehryar Mohri. 3-way composition of weighted finite-state transducers. In *International Conference on Implementation and Application of Automata*, pages 262–273. Springer, 2008.

Cyril Allauzen, Mehryar Mohri, and Brian Roark. Generalized algorithms for constructing statistical language models. In *Proceedings of the 41st Annual*

Meeting on Association for Computational Linguistics-Volume 1, pages 40–47. Association for Computational Linguistics, 2003.

Raman Arora, Ofer Dekel, and Ambuj Tewari. Online bandit learning against an adaptive adversary: from regret to policy regret. In *ICML*, 2012.

Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, 2002.

Robert J. Aumann. Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics*, 1(1):67–96, March 1974.

Borja Balle and Mehryar Mohri. Spectral learning of general weighted automata via constrained matrix completion. In *26th Annual Conference on Neural Information Processing Systems 2012, NIPS 2012*, 2012.

Borja Balle and Mehryar Mohri. On the Rademacher complexity of weighted automata. In *International Conference on Algorithmic Learning Theory*, pages 179–193. Springer, 2015.

Peter L. Bartlett and Shahar Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *JMLR*, 3, 2002.

Peter L. Bartlett, Elad Hazan, and Alexander Rakhlin. Adaptive online gradient descent. In *NIPS*, pages 65–72, 2007.

Omar Besbes, Yonatan Gur, and Assaf Zeevi. Stochastic multi-armed-bandit problem with non-stationary rewards. In *Advances in neural information processing systems*, pages 199–207, 2014.

Omar Besbes, Yonatan Gur, and Assaf Zeevi. Non-stationary stochastic optimization. *Operations Research*, 63(5):1227–1244, 2015.

Alina Beygelzimer, Elad Hazan, Satyen Kale, and Haipeng Luo. Online gradient boosting. In *Proceedings of NIPS*, pages 2449–2457, 2015a.

Alina Beygelzimer, Satyen Kale, and Haipeng Luo. Optimal and adaptive algorithms for online boosting. In *ICML*, volume 37 of *JMLR Proceedings*, 2015b.

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

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

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

Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, New York, NY, USA, 2006.

Nicolò Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, 2004.

Nicolò Cesa-Bianchi, Yishay Mansour, and Gilles Stoltz. Improved second-order bounds for prediction with expert advice. *Machine Learning*, 66(2-3):321–352, 2007.

Nicolò Cesa-Bianchi, Pierre Gaillard, Gábor Lugosi, and Gilles Stoltz. Mirror descent meets fixed share (and feels no regret). In *Advances in Neural Information Processing Systems*, pages 980–988, 2012.

Nicolò Cesa-Bianchi, Ofer Dekel, and Ohad Shamir. Online learning with switching costs and other adaptive adversaries. In *NIPS*, pages 1160–1168, 2013.

Stanley Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. Technical Report, TR-10-98, Harvard University, 1998.

Chao-Kai Chiang, Tianbao Yang, Chia-Jung Lee, Mehrdad Mahdavi, Chi-Jen Lu, Rong Jin, and Shenghuo Zhu. Online optimization with gradual variations. In *COLT*, pages 6.1–6.20, 2012.

Chao-Kai Chiang, Chia-Jung Lee, and Chi-Jen Lu. Beating bandits in gradually evolving worlds. In *COLT*, pages 210–227, 2013.

Corinna Cortes, Mehryar Mohri, and Umar Syed. Deep boosting. In *Proceedings of ICML*, 2014.

Maxime Crochemore. Transductions and repetitions. *Theoretical Computer Science*, 45:63–86, 1986.

Amit Daniely, Alon Gonen, and Shai Shalev-Shwartz. Strongly adaptive online learning. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1405–1411, 2015.

Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, pages 271–280. ACM, 2007.

Ofer Dekel and Yoram Singer. Data-driven online to batch conversions. In *NIPS*, pages 267–274, 2005.

Thomas G Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, 40(2):139–157, 2000.

John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. In *COLT*, pages 257–269, 2010.

Francoise Forges. An approach to communication equilibria. *Econometrica*, 54(6): pp. 1375–1385, 1986.

Dean P. Foster and Rakesh V. Vohra. Calibrated learning and correlated equilibrium. *Games and Economic Behavior*, 21(12):40 – 55, 1997.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer 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 *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 334–343. ACM, 1997.

Amy Greenwald, Zheng Li, and Warren Schudy. More efficient internal-regret-minimizing algorithms. In *COLT*, pages 239–250. Omnipress, 2008.

András György and Csaba Szepesvári. Shifting regret, mirror descent, and matrices. In *ICML*, 2016.

András Gyorgy, Tamás Linder, and Gábor Lugosi. Efficient tracking of large classes of experts. *IEEE Transactions on Information Theory*, 58(11):6709–6725, 2012.

N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2):217–288, 2011. ISSN 0036-1445.

Eric C Hall and Rebecca M Willett. Online optimization in dynamic environments. *arXiv preprint arXiv:1307.5944*, 2013.

Elad Hazan and Satyen Kale. Better algorithms for benign bandits. In *SODA*, pages 38–47, 2009.

Elad Hazan and Comandur Seshadhri. Efficient learning algorithms for changing environments. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 393–400. ACM, 2009.

Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, pages 1–9. ACM, 2014.

Mark Herbster and Manfred K Warmuth. Tracking the best expert. *Machine Learning*, 32(2):151–178, 1998.

Daniel Hsu, Sham M Kakade, and Tong Zhang. A spectral algorithm for learning hidden markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480, 2012.

Ali Jadbabaie, Alexander Rakhlin, Shahin Shahrampour, and Karthik Sridharan. Online optimization: Competing with dynamic comparators. In *AISTATS*, 2015.

Frederick Jelinek and Robert L. Mercer. Interpolated estimation of markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, pages 381–397, 1980.

Rong Jin, Steven CH Hoi, and Tianbao Yang. Online multiple kernel learning: Algorithms and mistake bounds. In *International Conference on Algorithmic Learning Theory*, pages 390–404. Springer, 2010.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *NIPS*, pages 315–323, 2013.

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

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

Slava M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recogniser. *IEEE Transactions on Acoustic, Speech, and Signal Processing*, 35(3):400–401, 1987.

Jyrki Kivinen and Manfred K Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

Robert Kleinberg, Alexandru Niculescu-Mizil, and Yogeshwer Sharma. Regret

bounds for sleeping experts and bandits. *Machine learning*, 80(2-3):245–272, 2010.

D.E. Knuth, J.H. Morris Jr, and V.R. Pratt. Fast pattern matching in strings. *SIAM Journal of Comput. Syst. Sci.*, 6:323–350, 1977.

Vladimir Koltchinskii and Dmitry Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *Annals of Statistics*, pages 1–50, 2002.

Wouter M Koolen and Steven de Rooij. Universal codes from switching strategies. *IEEE Transactions on Information Theory*, 59(11):7168–7185, 2013.

Ehud Lehrer. Correlated equilibria in two-player repeated games with nonobservable actions. *Mathematics of Operations Research*, 17(1):pp. 175–199, 1992.

Nick Littlestone. From on-line to batch learning. In *COLT*, pages 269–284, 1989.

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

H. Brendan McMahan. Analysis techniques for adaptive online learning. *CoRR*, 2014.

H. Brendan McMahan and Matthew J. Streeter. Adaptive bound optimization for online convex optimization. In *COLT*, pages 244–256, 2010.

H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM*

SIGKDD international conference on Knowledge discovery and data mining, pages 1222–1230. ACM, 2013.

Mehryar Mohri. String-Matching with Automata. *Nordic Journal of Computing*, 4 (2):217–231, Summer 1997.

Mehryar Mohri. Weighted automata algorithms. In *Handbook of weighted automata*, pages 213–254. Springer, 2009.

Mehryar Mohri and Mark-Jan Nederhof. Regular approximation of context-free grammars through transformation. In *Robustness in language and speech technology*, pages 153–163. Springer, 2001.

Mehryar Mohri and Scott Yang. Conditional swap regret and conditional correlated equilibrium. In *Advances in Neural Information Processing Systems*, pages 1314–1322, 2014.

Mehryar Mohri and Scott Yang. Accelerating online convex optimization via adaptive prediction. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 848–856, 2016a.

Mehryar Mohri and Scott Yang. Structural online learning. In *International Conference on Algorithmic Learning Theory*, pages 223–237. Springer, 2016b.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.

Aryan Mokhtari, Shahin Shahrampour, Ali Jadbabaie, and Alejandro Ribeiro. Online optimization in dynamic environments: Improved regret rates for strongly

convex problems. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 7195–7201. IEEE, 2016.

Claire Monteleoni and Tommi S Jaakkola. Online learning of non-stationary sequences. In *Advances in Neural Information Processing Systems*, 2003.

Claire Monteleoni, Gavin A Schmidt, and Scott McQuade. Climate informatics: accelerating discovering in climate science with machine learning. *Computing in Science & Engineering*, 15(5):32–40, 2013.

Mark-Jan Nederhof. Practical experiments with regular approximation of context-free languages. *Computational Linguistics*, 26(1):17–44, 2000.

Yurii Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, pages 341–362, 2012.

Hermann Ney, Ute Essen, and Reinhard Kneser. On structuring probabilistic dependences in stochastic language modeling. *Computer Speech and Language*, 8: 1–38, 1994.

Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA, 2007.

Francesco Orabona, Koby Crammer, and Nicolò Cesa-Bianchi. A generalized online mirror descent with applications to classification and regression. *Machine Learning*, 99(3):411–435, 2015.

Fernando CN Pereira and Rebecca N Wright. Finite-state approximation of phrase structure grammars. In *Proceedings of the 29th annual meeting on Association*

for Computational Linguistics, pages 246–255. Association for Computational Linguistics, 1991.

Leonard Pitt and Manfred K Warmuth. The minimum consistent DFA problem cannot be approximated within any polynomial. *Journal of the ACM (JACM)*, 40(1):95–142, 1993.

Alexander Rakhlin and Karthik Sridharan. Online learning with predictable sequences. In *COLT*, pages 993–1019, 2013a.

Alexander Rakhlin and Karthik Sridharan. Optimization, learning, and games with predictable sequences. In *NIPS*, pages 3066–3074, 2013b.

Alexander Rakhlin, Karthik Sridharan, and Ambuj Tewari. Online learning: Random averages, combinatorial parameters, and learnability. In *Proceedings of NIPS*, pages 1984–1992, 2010.

Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Relax and randomize : From value to algorithms. In *NIPS*, pages 2150–2158, 2012.

Alexander Rakhlin, Karthik Sridharan, and Ambuj Tewari. Online learning via sequential complexities. *Journal of Machine Learning Research*, 16:155–186, 2015.

Gunnar Rätsch, Takashi Onoda, and K-R Müller. Soft margins for adaboost. *Machine learning*, 42(3):287–320, 2001.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 693–701, 2011.

Alfréd Rényi et al. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. The Regents of the University of California, 1961.

R. Tyrell Rockafellar. *Convex analysis*. Princeton University Press, 1970.

Ivan Nikolaevich Sanov. On the probability of large deviations of random magnitudes. *Matematicheskii Sbornik*, 84(1):11–44, 1957.

Shahin Shahrampour and Ali Jadbabaie. Distributed online optimization in dynamic environments using mirror descent. *arXiv preprint arXiv:1609.02845*, 2016.

Shai Shalev-Shwartz. Online learning and online convex optimization. *Found. Trends Mach. Learn.*, pages 107–194, 2012.

Shai Shalev-Shwartz and Ambuj Tewari. Stochastic methods for l1-regularized loss minimization. *Journal of Machine Learning Research*, pages 1865–1892, 2011.

Jacob Steinhardt and Percy Liang. Adaptivity and optimism: An improved exponentiated gradient algorithm. In *ICML*, pages 1593–1601, 2014.

Gilles Stoltz and Gábor Lugosi. Learning correlated equilibria in games with compact sets of strategies. *Games and Economic Behavior*, 59(1):187–208, 2007.

Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. SIAM: Society for Industrial and Applied Mathematics, 1997.

Tim Van Erven and Peter Harremos. Rényi divergence and kullback-leibler divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820, 2014.

VN Vapnik. Principles of risk minimization for learning theory. *Advances in Neural Information Processing Systems*, 4:831–838, 1992.

Vladimir Vovk. Derandomizing stochastic prediction strategies. *Machine Learning*, 35(3):247–282, 1999.

Chen-Yu Wei, Yi-Te Hong, and Chi-Jen Lu. Tracking the best expert in non-stationary stochastic environments. In *Advances in neural information processing systems*, 2016.

Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1–9, 2015.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, pages 928–936, 2003.