Guarantees for Efficient and Adaptive Online Learning

James David Robinson

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
of
University College London.

Department of Computer Science
University College London

June 1, 2023

I, James David Robinson, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

In this thesis, we study the problem of adaptive online learning in several different settings. We first study the problem of predicting graph labelings online which are assumed to change over time. We develop the machinery of *cluster specialists* which probabilistically exploit any cluster structure in the graph. We give a mistake-bounded algorithm that surprisingly requires only $\mathcal{O}(\log n)$ time per trial for an n-vertex graph, an exponential improvement over existing methods.

We then consider the model of non-stationary prediction with expert advice with long-term memory guarantees in the sense of Bousquet and Warmuth [1], in which we learn a small pool of experts. We consider relative entropy projection-based algorithms, giving a linear-time algorithm that improves on the best known regret bound [2]. We show that such projection updates may be advantageous over previous "weight-sharing" approaches when weight updates come with implicit costs such as in portfolio optimization. We give an algorithm to compute the relative entropy projection onto the simplex with non-uniform (lower) box constraints in linear time, which may be of independent interest.

We finally extend the model of long-term memory by introducing a new model of adaptive long-term memory. Here the small pool is assumed to change over time, with the trial sequence being partitioned into *epochs* and a small pool associated with each epoch. We give an efficient linear-time regret-bounded algorithm for this setting and present results in the setting of contextual bandits.

Impact Statement

The work contained in this thesis has the following potential benefits both inside and outside academia. Firstly, this thesis contains several novel algorithms developed for a variety of machine learning problems, with worst-case performance guarantees. These algorithms and the ideas therein may form the basis for future research and improvements in online machine learning. Furthermore, while this thesis is presented in the context of online machine learning, some of the ideas are much more general and may be of independent interest. For example, the machinery of cluster specialists developed in Chapter 3 for predicting on graphs is quite general and could have many uses outside of online learning. As another example, we give an algorithm to compute the relative entropy projection onto the simplex with non-uniform box constraints in linear time which may have quite general applicability.

Regarding the potential benefits outside of academia, while this research in adaptive online learning is theoretical, we do include experiments on real-world data (Chicago bicycle sharing data). Our results show promising algorithms for predicting the states of bicycle-sharing stations around a city as they evolve over time. This is a small example of how such algorithms could be used in city transport optimization, or more generally for public safety.

Finally, we give a brief reflection on the ethical considerations of this work. While the scope of applicability of online learning algorithms is wide, this research in regret-bounded online learning is foundational in nature and we therefore cannot foresee the extent of any societal impacts (positive or negative) this research may have.

Acknowledgements

First and foremost, I would like to thank (with no hope of fully expressing the depth of my gratitude through these words alone) my supervisor, Mark Herbster. Thank you, Mark, for your time, patience, wisdom, and friendship. Thank you for teaching me how to do research, and how to approach and think about a problem. I could not have hoped for a better supervisor during my PhD, and I am very grateful.

I would like to give my thanks to several colleagues at UCL. Especially to my second supervisor, Massimiliano Pontil, for sharing with me your passion for problem-solving. Thank you to Stephen Pasteris for infecting me with me your enthusiasm for online learning, and making me appreciate the power of binary trees. Thank you to Tim Law for the long conversations on switching with memory, and portfolio selection. Thank you to Dmitry Adamskiy for your helpful insights and discussions on specialists. Thank you to Lisa Tse and Pawel Chilinski for your helpful advice and comments.

I would also like to thank my family for their support and encouragement during all of my educational years.

Last but not least, thank you to my wife, Dain, for your endless patience and support during my PhD. I could not have done this without you.

I would also like to express my gratitude for my funding. This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/N509577/1.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The impact that the field of machine learning and artificial intelligence has had on the world in recent years is hard to overstate. From supercomputers to the phones in our pockets, machine learning techniques and algorithms are increasingly used in our lives. This use of machine learning methods would not be possible without the enormous amounts of data that are captured around the world every day.

The field of machine learning can be broadly split into two camps, *batch* learning and *online* learning. In batch learning, a complete dataset is used to train a predictor, after which predictions are made. Conversely, in online learning, data arrives sequentially and a predictor must make predictions at each step or trial and learn incrementally. Online learning algorithms typically have a small memory footprint and often do not require one to store data. As such, with the vast amount of data being generated and recorded today, online learning algorithms and techniques are becoming increasingly important. Furthermore, the world is in a constant state of flux. From seasonal changes in the weather to the mass movements of people throughout a city every day, an attractive property of many online learning algorithms is their ability to adapt to change and continue to predict accurately.

In this thesis, we develop and study several online learning algorithms in

various settings. A core feature of these algorithms is their *adaptability*. That is, we assume that the nature of the data on which they make predictions is changing over time, and the algorithms must adapt accordingly. Throughout this thesis, we will also have a secondary focus on *efficiency*. While "efficiency" will refer to the computational performance of the algorithms studied (in both time and space complexity) for most of this thesis, the word will occasionally be used in a different context to compare different algorithms.

1.2 Contributions

The main original contributions of this thesis are as follows.

1.2.1 Online Learning on Graphs

We develop the machinery of cluster specialists for the problem of predicting graph labelings. Cluster specialists probabilistically exploit the cluster structure in the graph. We present an algorithm for using cluster specialists to predict switching graph labelings online. Our proposed algorithm has two variants, of which one surprisingly only requires $\mathcal{O}(\log n)$ time on any trial on an n-vertex graph, an exponential speed-up over existing methods. We prove switching mistake-bound guarantees for both variants of our cluster specialist algorithm.

1.2.2 Online Learning with Expert Advice

We present an $\mathcal{O}(n)$-time per trial projection-based algorithm for which we prove the best known regret bound for tracking experts with memory. We show that this projection-based algorithm is intimately related to a more traditional "weight-sharing" algorithm, which we show is a new method for *Mixing Past Posteriors* (MPP) [1]. We show that this method surprisingly corresponds to the algorithm with the previous best known regret bound for this problem [2]. We give an efficient $\mathcal{O}(n)$-time algorithm for computing exact relative entropy projection onto a simplex with non-uniform (lower) box constraints. We provide a guarantee which favors projection-based updates over weight-sharing updates when updating weights may incur costs.

1.2.3 Adaptive Long-Term Memory

We introduce a refined model for "switching with memory," which we study in both full-information and partial-information settings. We present results in the setting of contextual bandits. We develop an algorithm that is capable of learning an "adaptive" small pool (as in the switching with memory setting) that is changing over time. Our algorithm requires only $\mathcal{O}(n)$ time per trial, where n is the number of policies (experts).

1.3 Publications

The results of Chapter 3 on online prediction of switching graph labelings were presented at NeurIPS 2019 and published in the proceedings:

- Mark Herbster and James Robinson. Online prediction of switching graph labelings with cluster specialists. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

The work was done in collaboration with Mark Herbster.
The results of Chapter 4 on improved regret bounds for tracking experts with memory were presented at NeurIPS 2021 and published in the proceedings:

- James Robinson and Mark Herbster. Improved regret bounds for tracking experts with memory. In *Advances in Neural Information Processing Systems*, volume 34, pages 7625–7636, 2021.

The work was done in collaboration with Mark Herbster.
At the time of writing the results of Chapter 5 on adaptive long-term memory are to be submitted for peer review and publication:

- Stephen Pasteris, James Robinson, Massimiliano Pontil, and Mark Herbster. Adaptive long-term memory for experts and bandits. Submitted for review, 2023.

In that work we present results and an algorithm for the combined settings of adaptive long-term memory and multitask learning. These results are

independent. The multitask setting is out of the scope of this thesis, and we therefore present results on adaptive long-term memory only (that is, for the "single task" setting). The work was done in collaboration with Stephen Pasteris, Mark Herbster, and Massimiliano Pontil.

1.4 Thesis Structure

The content of this thesis is organized as follows. In Chapter 2, we give a gentle introduction to the general online learning protocol studied in this thesis. We then give a brief overview of the various online learning problems studied in each chapter of this thesis, including the problems of online prediction on graphs, prediction with expert advice, and prediction with partial feedback. In Chapter 3, we present our main results on predicting a switching sequence of graph labelings, including experiments. In Chapter 4, we present our main results on tracking a small pool of experts using projection-based algorithms. In Chapter 5 we present our main results on adaptive long-term memory, including experiments. Finally, in Chapter 6 we give some concluding remarks.

Chapter 2

Background

In this chapter, we give some background into the various online learning settings considered in this thesis. Since each chapter of this thesis is somewhat self-contained, we give a more specific and detailed background and review of the relevant literature in each chapter.

2.1 Online Learning

We first introduce the online learning protocol. In this protocol, learning proceeds in trials. Let $\mathcal{Y}$ be an outcome space, and let $y^1, y^2, \ldots, y^T \in \mathcal{Y}$ be a sequence of outcomes incrementally revealed to a `learner` over T trials by `nature`. Before the `learner` observes the outcome y^t on trial t however, they must make a prediction $\hat{y}^t \in \mathcal{D}$, where $\mathcal{D}$ is a prediction space. We take the pessimistic view by focusing on the failures of the `learner` rather than its successes by defining a non-negative loss function $\ell : \mathcal{D} \times \mathcal{Y} \to [0, \infty]$.

Note that we may be required to distinguish between $\mathcal{D}$ and $\mathcal{Y}$, since although there are many natural problems where $\mathcal{Y} = \mathcal{D}$, they may be different. For example, if we were to predict whether or not it will snow tomorrow, we would have $\mathcal{Y} = \{0, 1\}$, and might require our algorithm to output the *probability* of it snowing tomorrow, and thus $\mathcal{D} = [0, 1]$. The `learner` may also receive some side information, $\boldsymbol{x}^t \in \mathcal{X}$, such as predictions of the probability of snow tomorrow from n weather stations, in which case $\mathcal{X} = \mathcal{D}^n$. The general online learning protocol is given in Protocol 1. Usually, no statistical

Protocol 1 The Online Learning Protocol

for $t \leftarrow 1$ to T **do**
  `Learner` receives $\boldsymbol{x}^t \in \mathcal{X}$
  `Learner` predicts $\hat{y}^t \in \mathcal{D}$
  `Nature` observes $y^t \in \mathcal{Y}$
  `Learner` receives $\ell(\hat{y}^t, y^t)$
end for

assumptions are made about the nature of the data, and thus the online learning protocol can be viewed as a game between the `learner` and `nature`, which may very well be adversarial.

As an example, a common loss function used in regression problems is the *square-loss*, given by $\ell(\hat{y}, y) = (\hat{y} - y)^2$, in which case we usually have $\mathcal{D} = \mathcal{Y} = \mathbb{R}$ or $\mathcal{D} = \mathbb{R}$ and $\mathcal{Y} = [-Y, Y]$ for some positive constant Y.

Another example is the logarithmic loss, for prediction space $\mathcal{D} = [0, 1]$ and outcome space $\{0, 1\}$:

$$\ell(\hat{y}, y) = \begin{cases} -\log(\hat{y}), & y = 1 \\ -\log(1 - \hat{y}), & y = 0, . \end{cases}$$

As a final example consider the case that $\mathcal{D} = \{0, 1\}$, $\mathcal{Y} = \{0, 1\}$, and $\ell(\hat{y}, y) = [\![\hat{y} \neq y]\!]$, where $[\![\texttt{pred}]\!]$ is equal to 1 if the predicate `pred` is true and 0 otherwise. This setting defines the mistake-bound model, first introduced by Littlestone [6]. Note that the total loss accumulated by the `learner` over the T trials coincides with the number of mistakes made. In Chapter 3, we present results within this mistake-counting setting.

2.2 Online Learning with Expert Advice

Central to this thesis is the classic problem of online prediction with expert advice [7, 8, 9]. In this model, before making each prediction, the `learner` listens to a set of n experts who each make their own predictions. Thus the `learner` receives $\boldsymbol{x}^t \in \mathcal{D}^n$. These experts may be machine learning algorithms,

Protocol 2 The Prediction with Expert Advice Protocol

for $t \leftarrow 1$ to T **do**
  `Learner` receives expert predictions $\boldsymbol{x}^t \in \mathcal{D}^n$
  `Learner` predicts $\hat{y}^t \in \mathcal{D}$
  `Nature` reveals $y^t \in \mathcal{Y}$
  `Learner` receives $\ell^t = \ell(\hat{y}^t, y^t)$
  Expert i receives $\ell_i^t = \ell(x_i^t, y^t)$ for $i \in [n]$
end for

humans, or even virtual constructions. The `learner` bases its prediction on the advice of the experts. After the prediction is made and the true outcome is revealed by `nature`, the `learner` and all experts each incur a loss, measured by the loss function. In the case that the `learner` receives information on all expert losses on each trial, we call this the *full-information* setting. Later we discuss the setting where the `learner` receives only partial feedback. The set of experts may be infinite, but for simplicity we will assume for now that it is finite. The protocol for prediction with expert advice is given in Protocol 2. Note that we have introduced $\ell^t := \ell(\hat{y}^t, y^t)$ and $\ell_i^t := \ell(x_i^t, y^t)$ (the loss of expert i) for convenience.

Recall that we make no statistical assumptions on the nature of how the data is generated, indeed we assume that it may be generated *adversarially* by `nature`. Of course, if nature is truly adversarial, then the `learner` has little hope of performing well since it can incur a maximal loss on every trial. The goal of the `learner` is then to predict well relative to a predetermined comparison class of predictors. In the case of prediction with expert advice, this is the set of experts themselves. For a given data sequence, we define the *regret* of the `learner` with respect to expert i as

$$\mathcal{R}(i) := \sum_{t=1}^{T} \ell^t - \sum_{t=1}^{T} \ell_i^t \,.$$

We call this the standard regret model. This measure of regret relative to a fixed expert, i, is of course very restrictive. An extension to this model, and

central to this thesis, is the notion of "switches" in the data, such that different experts may perform well at different times. In this setting, the performance of the `learner` is compared to the best *sequence* of experts. For any comparison sequence of experts $i_{1:T} = i_1, \ldots, i_T \in [n]$ the regret of the `learner` with respect to this sequence is defined as

$$\mathcal{R}(i_{1:T}) := \sum_{t=1}^{T} \ell^t - \sum_{t=1}^{T} \ell^t_{i_t} \,. \tag{2.1}$$

Our goal in this thesis will be to derive efficient algorithms with good regret bound guarantees for such settings.

2.2.1 Specialists

An extension to the problem of learning with expert advice is that of *specialists*. First introduced in [10], specialists are defined as experts who can abstain from predicting on any given trial. Thus the `learner` may now receive predictions $\boldsymbol{x}^t \in (\mathcal{D} \cup \{\square\})^n$, where $\square$ represents an abstention. A specialist that abstains is said to be *asleep*, while a specialist that offers a prediction is said to be *awake*.

Specialist algorithms have proved to be powerful tools for developing adaptive online learning algorithms [2, 11, 12], and we make use of the notion of specialists in this thesis in Chapter 3 and Chapter 5.

2.3 Online Learning on Graphs

In Chapter 3, we consider the problem of predicting binary labelings of a graph. Let $\mathcal{G} = (V, E)$ be an undirected, connected, n-vertex graph with vertex set V, and edge set E. Consider Protocol 3 which describes the `learner` predicting a binary labeling of a graph in the online setting.

Note that here we have defined $m_t := \ell(\hat{y}^t, y^t) = [\![\hat{y}^t \neq y^t]\!]$. In this simple game, the goal of `learner` is to minimize the total number of mistakes,

$$\sum_{t=1}^{T} m_t = |\{t : \hat{y}^t \neq y^t\}| \,.$$

Protocol 3 The Binary Graph Label Prediction Protocol

`Nature` presents a graph $\mathcal{G} = (V, E)$
for $t \leftarrow 1$ to T **do**
  `Nature` queries a vertex $i_t \in V = \{1, 2, \ldots, n\}$
  `Learner` predicts the label $\hat{y}^t \in \{-1, 1\}$ of i_t
  `Nature` presents the true label $y^t \in \{-1, 1\}$
  `Learner` suffers loss $m_t = [\![\hat{y}^t \neq y^t]\!]$
end for

Again, we do not assume that there is a fixed labeling on the graph, but instead that this labeling is changing over time, and to do well our algorithm must adapt to these changes in labelings. Consider an example of services placed throughout a city, such as public bicycle sharing stations. As the population uses these services the state of each station–such as the number of available bikes–naturally evolves throughout the day, at times gradually and others abruptly, and we might want to predict the next state of any given station at any given time. Since the location of a given station as well as the state of nearby stations will be relevant to this learning problem, it is natural to use a graph-based approach.

Another setting might be a graph of major road junctions (vertices) connected by roads (edges), in which one wants to predict whether or not a junction is congested at any given time. Traffic congestion is naturally non-stationary and also exhibits both gradual and abrupt changes to the structure of the labeling over time [13].

Both of these examples demonstrate real-world instances of Protocol 3. The goal of Chapter 3 is to develop *efficient* algorithms–as well as their mistake bound guarantees–that can predict the labelings of these graphs online when they are changing over time.

2.4 Partial Information

We now introduce two further variants of Protocol 2 which have proven to be very useful in many real-world settings [14]. The first variant is the notion of an

Protocol 4 The Partial-Information Protocol

for $t \leftarrow 1$ to T **do**
  `Learner` chooses $\boldsymbol{w}^t \in \Delta_A$
  `Learner` selects action $a^t \sim \boldsymbol{w}^t$
  `Nature` reveals $\ell^t_{a^t} \in [0, 1]$
  `Learner` receives $\ell^t = \ell^t_{a^t}$
end for

action rather than a prediction. Instead of the `learner` predicting with $\hat{y}^t \in \mathcal{D}$, the learner instead maintains a weight distribution over a set of A actions, and on each trial samples an action from this is distribution. The second variant is the notion of partial information (or *bandits*). In this setting, on each trial the `learner` only receives feedback from `nature` about the selected action on that trial. This setting is given in Protocol 4, where we assume that losses take values in $[0, 1]$. Since the `learner` does not know on any given trial how it would have faired by selecting an alternative action, it must thus try to find a balance between *exploration* (finding good actions) and *exploitation* (playing actions that appear to be good). See [14] for an overview and history into this problem.

In Chapter 5 we will study a particular instance of the partial information problem known as adversarial *contextual bandits*. Closely related to the problem of prediction with expert advice, in the contextual bandits setting, `nature` presents a context vector to the `learner` who has access to a number of *policies* which, given this context vector, output distributions over the available actions. The `learner` bases its prediction on the advice of the policies, and our (expected) regret is measured with respect to the best policy, or sequence of policies in the non-stationary case. We define this setting more formally in Chapter 5.

2.5 Thesis Focus

In this chapter we gave a brief introduction to a variety of topics and protocols in online machine learning. The focus of this thesis is on the development of *efficient* adaptive algorithms and proving good performance guarantees in these settings. While we cover a variety of problems, the techniques used and developed in this work all have roots in prediction with expert advice and the extensions introduced previously.

2.6 Notation

We finally introduce some notation that is used across this thesis. Where a chapter contains notation specific to that chapter only, it is defined in place.

We define $\mathbb{N}$ to be the set of natural numbers (not including zero) and for all $n \in \mathbb{N}$ we define $[n] := \{1, \dots, n\}$. Given a set X and some $n \in \mathbb{N}$, an n-dimensional vector over X is a sequence of n elements of X. Given $\boldsymbol{x}$, an n-dimensional vector, the i-th component of $\boldsymbol{x}$ is denoted x_i. The set of all n-dimensional vectors over X is denoted X^n. Let $\Delta_n := \{\boldsymbol{u} \in [0,1]^n : \|\boldsymbol{u}\|_1 = 1\}$ be the $(n-1)$-dimensional probability simplex. Let $\mathbf{1}$ denote the vector $(1, \dots, 1)$ and $\mathbf{0}$ denote the vector $(0, \dots, 0)$. Let $\boldsymbol{e}_i$ denote the i^{th} standard basis vector. We define $D(\boldsymbol{u}, \boldsymbol{w}) := \sum_{i=1}^{n} u_i \log \frac{u_i}{w_i}$ to be the relative entropy between $\boldsymbol{u}$ and $\boldsymbol{w}$. We denote component-wise multiplication as $\boldsymbol{u} \odot \boldsymbol{w} := (u_1 w_1, \dots, u_n w_n)$. For $p \in [0,1]$ we define $\mathcal{H}(p) := -p \ln p - (1-p) \ln (1-p)$ to be the binary entropy of p, using the convention that $0 \ln 0 = 0$.

Chapter 3

Online Prediction of Switching Graph Labelings

3.1 Introduction to the Chapter

In this chapter we address the problem of predicting the labeling of a graph online when the labeling is changing over time. We briefly introduced the model of predicting a binary labeling online in Section 2.3. Recall that in this model, on every trial `nature` queries a vertex in the graph, for which the `learner` predicts a binary label. `Nature` then reveals the true label, and the `learner` is either correct or incurs a mistake. We will assume in our model that the labeling chosen by `nature` is changing over time, and our algorithm must adapt to this change in labeling.

In this chapter we present an algorithm based on a specialist [10] approach; we develop the machinery of cluster specialists, which probabilistically exploits the cluster structure in the graph. The chapter is organized as follows. We first introduce some additional notation required for this chapter only. We then introduce the model in Section 3.2 and review the relevant literature. In Section 3.4 we present the Switching Cluster Specialists algorithm (SCS), a modification of the method of specialists [10] with the novel machinery of *cluster specialists*, a set of specialists that, in a rough sense, correspond to clusters in the graph. We consider two distinct sets of specialists, $\mathcal{B}_n$ and $\mathcal{F}_n$,

where $\mathcal{B}_n \subset \mathcal{F}_n$. With the smaller set of specialists, $\mathcal{B}_n$, the bound is only larger by a factor of $\log n$. On the other hand, we will see that prediction is exponentially faster per trial, remarkably requiring only $\mathcal{O}(\log n)$ time to predict. In Section 3.7 we present results of experiments on Chicago Divvy Bicycle Sharing data.

3.1.1 Notation

We first introduce some additional notation specific to this chapter. Let $\mathcal{G} = (V, E)$ be an undirected, connected, n-vertex graph with vertex set $V = \{1, 2, \ldots, n\}$ and edge set E. Each vertex of this graph may be labeled with one of two states $\{-1, 1\}$ and thus a labeling of a graph may be denoted by a vector $\boldsymbol{\mu} \in \{-1, 1\}^n$ where μ_i denotes the label of vertex i. The underlying assumption in our model is that we are predicting vertex labels from a sequence $\boldsymbol{\mu}^1, \ldots, \boldsymbol{\mu}^T \in \{-1, 1\}^n$ of graph labelings over T trials. The set

$$\mathcal{K} := \{t \in \{2, \ldots, T\} : \boldsymbol{\mu}^t \neq \boldsymbol{\mu}^{t-1}\} \cup \{1\}$$

contains the first trial of each of the $|\mathcal{K}|$ "segments" of the prediction problem. Each segment corresponds to a time period when the underlying labeling is unchanging. The *cut-size* of a labeling $\boldsymbol{\mu}$ on a graph $\mathcal{G}$ is defined as

$$\Phi_{\mathcal{G}}(\boldsymbol{\mu}) := \left|\{(i, j) \in E : \mu_i \neq \mu_j\}\right|,$$

i.e., the number of edges between vertices of disagreeing labels.

We let $r_{\mathcal{G}}(i, j)$ denote the *resistance distance* (effective resistance) between vertices i and j when the graph $\mathcal{G}$ is seen as a circuit where each edge has unit resistance (e.g., [15]). The effective resistance for an unweighted graph $\mathcal{G}$ can be written as

$$r_{\mathcal{G}}(i, j) = \frac{1}{\min\limits_{\boldsymbol{\mu} \in \mathbb{R}^n} \{ \sum\limits_{(p,q) \in E} (\mu_p - \mu_q)^2 : \mu_i = 0, \mu_j = 1\}}$$

The *resistance diameter* of a graph is $R_{\mathcal{G}} := \max_{i,j \in V} r_{\mathcal{G}}(i,j)$. The *resistance weighted* cut-size of a labeling $\boldsymbol{\mu}$ is

$$\Phi^r_{\mathcal{G}}(\boldsymbol{\mu}) := \sum_{(i,j) \in E : \mu_i \neq \mu_j} r_{\mathcal{G}}(i,j)\,.$$

For $\boldsymbol{u} \in \Delta_n$ we define $H(\boldsymbol{u}) := \sum_{i=1}^n u_i \log_2 \frac{1}{u_i}$ to be the entropy of $\boldsymbol{u}$. In this chapter, for $\boldsymbol{u}, \boldsymbol{w} \in \Delta_n$ we define $D(\boldsymbol{u}, \boldsymbol{w}) = \sum_{i=1}^n u_i \log_2 \frac{u_i}{w_i}$ to be the relative entropy between $\boldsymbol{u}$ and $\boldsymbol{w}$ with base 2. For a vector $\boldsymbol{w}$ and a set of indices $\mathcal{I}$ let $\boldsymbol{w}(\mathcal{I}) := \sum_{i \in \mathcal{I}} w_i$.

3.2 Background and Related Work

As discussed, if `nature` is strictly adversarial in choosing its labelings, then the `learner` can easily incur a mistake on every trial, but if `nature` is regular or simple, then there is hope that the `learner` may incur only a few mistakes. Thus, a central goal of mistake-bounded online learning is to design algorithms whose total number of mistakes can be bounded relative to the complexity of `nature`'s labeling of the graph. This (non-switching) graph labeling problem has been studied extensively in the online learning literature [16, 17, 18, 19, 20].

The problem of predicting the labeling of a graph in the batch setting was introduced as a foundational method for semi-supervised (transductive) learning. In this setting, the graph was built using both the unlabeled and labeled instances. The seminal work by [21] used a metric on the instance space and then built a kNN or ϵ-ball graph. The partial labeling was then extended to the complete graph by solving a mincut-maxflow problem where opposing binary labels represented sources and sinks. In practice, this method suffered from very unbalanced cuts. Significant practical and theoretical advances were made by replacing the mincut/maxflow model with methods based on minimizing a quadratic form of the graph Laplacian. Influential early results include but are not limited to [22, 23, 24]. A limitation of the graph Laplacian-based techniques is that these batch methods–depending on their implementation–

typically require $\Theta(n^2)$ to $\Theta(n^3)$ time to produce a single set of predictions.

Predicting the labeling of a graph in the online setting was introduced by [25]. The authors proved bounds for a Perceptron-like algorithm with a kernel based on the graph Laplacian. Common to all of these papers is that a dominant term in their mistake bounds is the (resistance-weighted) cut-size. The Perceptron-like algorithm of [25] with kernel $\boldsymbol{K} := \boldsymbol{L}^+$ (the pseudo-inverse of the graph Laplacian) has a mistake bound of

$$4\Phi_{\mathcal{G}}(\boldsymbol{\mu}) D_{\mathcal{G}} \mathrm{bal}(\boldsymbol{\mu}) \,, \tag{3.1}$$

where $D_{\mathcal{G}}$ is the diameter of the graph and $\mathrm{bal}(\boldsymbol{\mu}) := (1 - \frac{1}{n}|\sum_{i=1}^n \mu_i|)^{-2}$ is a measure of label balance in the graph. This label balance term makes the bound *very* sensitive, for example, a heavily imbalanced labeling (such as all but one vertices having the same label) gives $\mathrm{bal}(\boldsymbol{\mu}) = \mathcal{O}(n^2)$, causing the bound to be vacuous. Since this work, there has been a number of extensions and improvements in bounds including but not limited to [16, 17, 20, 26, 27, 28].

An explicit Kernel-Perceptron algorithm was presented in [29], with the following kernel

$$\boldsymbol{K}_c^b = \boldsymbol{L}^+ + b\mathbf{1}\mathbf{1}^\top + c\boldsymbol{I} \tag{3.2}$$

for $b > 0$, $c \geq 0$. The bound obtained by the Perceptron in the graph setting depends only on the cut-size and the diameter of the graph. Note that these terms are only indirectly related to the number of vertices in the graph. Concretely, in the realizable case for $b = 1$, $c = 0$, the mistake bound obtained when learning a labeling $\boldsymbol{\mu} \in \{-1, 1\}^n$ is then given by

$$\left(4\Phi_{\mathcal{G}}(\boldsymbol{\mu}) + \left(\frac{1}{n}\sum_{i=1}^n \mu_i\right)^2\right)(R_{\mathcal{G}} + 1) \tag{3.3}$$

where $R_{\mathcal{G}}$ is the resistance diameter of $\mathcal{G}$. In this chapter we consider an extension of this Kernel-Perceptron algorithm which allows switching, using it as a benchmark in our experiments.

From a simplified perspective, the methods for predicting the labeling of a graph (online) are split into two approaches. The first approach, as we have seen, works directly with the original graph and is usually based on a graph Laplacian [17, 20, 25]; it provides bounds that utilize the additional connectivity of non-tree graphs, which are particularly strong when the graph contains uniformly-labeled clusters of small (resistance) diameter. The drawbacks of this approach are that the bounds are weaker on graphs with large diameters, and that computation times are slower.

The second approach is to approximate the original graph with an appropriately selected tree or "line" graph [16, 18, 26, 19]. This enables faster computation times and bounds that are better on graphs with large diameters. These algorithms may be extended to non-tree graphs by first selecting a spanning tree uniformly at random [18] and then applying the algorithm to the sampled tree. This randomized approach induces *expected* mistake bounds that also exploit the cluster structure in the graph (see Section 3.3). We take this approach in this chapter.

In [16] the linearization of $\mathcal{G}$ to a so-called *spine*, $\mathcal{S}$, was introduced. A very exciting result was that a Bayesian treatment of predicting according to $P(\mu_{i_t} = y^t | \mu_{i_1} = y^1, \ldots, \mu_{i_{t-1}} = y^{t-1})$ was shown to be equivalent to predicting with the 1-nearest neighbor algorithm on $\mathcal{S}$, which can be done in $\mathcal{O}(n \log n)$ time. This led to a mistake bound of

$$2\Phi_{\mathcal{G}}(\boldsymbol{\mu}) \log_2 \left(\frac{n-1}{2\Phi_{\mathcal{G}}(\boldsymbol{\mu})} \right) + \frac{2\Phi_{\mathcal{G}}(\boldsymbol{\mu})}{\ln 2} + 1 \tag{3.4}$$

which is a vast improvement on graphs with a large diameter. We will also employ such a linearization technique in our method.

3.2.1 Switching Prediction

Recall that in this work our goal is to predict a (switching) sequence of graph labelings online. *Switching* in the mistake- or regret-bound setting refers to the problem of predicting an online sequence when the "best comparator" is

changing over time. In the simplest of switching models the set of comparators is *structureless*, and we simply pay per switch. A prominent early result in this model is [30] which introduced the *fixed-share* update, which will play a prominent role in our main algorithm. We will also study this update much more closely in Chapter 4 in the context of prediction with expert advice. Other prominent results in the structureless model include but are not limited to [1, 2, 31, 32, 33, 34]. A stronger model is to prove instead a bound that holds for any arbitrary contiguous sequence of trials. Such a bound is called an *adaptive-regret* bound. This type of bound automatically implies a bound on the structureless switching model. Adaptive-regret was introduced in [35][1] other prominent results in this model include [11, 34, 36].

The structureless model may be generalized by introducing a divergence measure on the set of comparators. Thus, whereas in the structureless model we pay for the number of switches, in the structured model we instead pay for the sum of divergences between successive comparators. This model was introduced in [37]; prominent results include [34, 38].

In [32] a variation of the *fixed-share* algorithm was given to allow certain exponentially large sets of comparators. In [35, 39] meta-algorithms were introduced with regret bounds which convert any "black box" online learning algorithm into an adaptive algorithm. Such methods could be used as an approach to predict switching graph labelings online, however these meta-algorithms introduce a factor of $\mathcal{O}(\log T)$ to the per-trial time complexity of the base online learning algorithm. In the online switching setting we will aim for our fastest algorithm to have $\mathcal{O}(\log n)$ time complexity per trial.

In this work we utilize specialists [10, 2, 11] by introducing a tailor-made set of specialists designed for the problem of predicting graph labelings. In [40] a similar approach of designing sets of specialists for the problems of predicting volatility of options and students' tests results was given.

In [27] the authors also consider switching graph label prediction. How-

[1]However, see the analysis of WML in [7] for a precursory result.

ever, their results are not directly comparable to ours since they consider the combinatorially more challenging problem of repeated switching within a small set of labelings contained in a larger set. That setup was a problem originally framed in the experts setting and posed as an open problem by [41] and solved in [1]. We also study this setting more closely in Chapter 4 in the experts setting. If we apply the bound in [27] to the case where there is *not* repeated switching within a smaller set, then their bound is uniformly and significantly weaker than the bounds in this paper and the algorithm is quite slow, requiring $\Theta(n^3)$ time per trial in a typical implementation. Also contained in [27] is a baseline algorithm based on a kernel perceptron with a graph Laplacian kernel. The bound of that algorithm has the significant drawback of scaling with the "worst" labeling in a sequence of labelings. However, it is simple to implement, and we use it as a benchmark in our experiments.

3.3 Random Spanning Trees and Linearization

Since we operate in the transductive setting where the entire unlabeled graph is presented to the `learner` beforehand, this affords the `learner` the ability to perform any reconfiguration to the graph as a preprocessing step. Recall that the bounds of most existing algorithms for predicting a labeling on a graph are usually expressed in terms of the cut-size of the graph under that labeling. A natural approach then is to use a spanning tree of the original graph which can only reduce the cut-size of the labeling.

The effective resistance between vertices i and j, denoted $r_{\mathcal{G}}(i,j)$, is equal to the probability that a spanning tree of $\mathcal{G}$ drawn uniformly at random (from the set of all spanning trees of $\mathcal{G}$) includes $(i,j) \in E$ as one of its $n-1$ edges (see e.g., [42]). As first observed by [26], by selecting a spanning tree uniformly at random from the set of all possible spanning trees, mistake bounds expressed in terms of the cut-size then become *expected* mistake bounds now in terms of the effective-resistance-weighted cut-size of the graph. That is, if $\mathcal{R}$ is a random spanning tree of $\mathcal{G}$ then $\mathbb{E}[\Phi_{\mathcal{R}}(\boldsymbol{\mu})] = \Phi^r_{\mathcal{G}}(\boldsymbol{\mu})$ and thus $\Phi^r_{\mathcal{G}}(\boldsymbol{\mu}) \leq \Phi_{\mathcal{G}}(\boldsymbol{\mu})$.

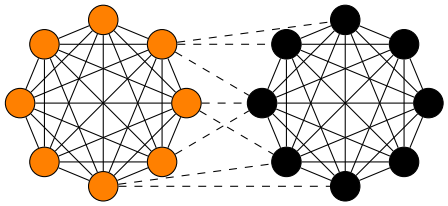


Figure 3.1: Two uniformly-labeled cliques, each of size m joined by $\ell \leq m$ edges. The cut-size of the given labeling is ℓ whereas the effective-resistance-weighted cut-size is $\mathcal{O}(1)$.

A random spanning tree can be sampled from a graph efficiently using a random walk or similar methods (see e.g., [43]).

To illustrate the power of this randomization, consider the simplified example of a graph with two cliques each of size $\frac{n}{2}$, where one clique is labeled uniformly with "+1" and the other "-1" with an additional arbitrary $\frac{n}{2}$ "cut" edges between the cliques (see Figure 3.1). This dense graph exhibits two disjoint clusters, and $\Phi_{\mathcal{G}}(\boldsymbol{\mu}) = \frac{n}{2}$. On the other hand $\Phi^r_{\mathcal{G}}(\boldsymbol{\mu}) = \Theta(1)$, since between any two vertices in the opposing cliques there are $\frac{n}{2}$ edge-disjoint paths of length ≤ 3 and thus the effective resistance between any pair of vertices is $\Theta(\frac{1}{n})$. Note that since bounds usually scale linearly with (resistance-weighted) cut-size, the cut-size bound would be vacuous but the resistance-weighted cut-size bound would be small.

We will make use of this preprocessing step of sampling a uniform random spanning tree, as well as a *linearization* of this tree to produce a (spine) line-graph, $\mathcal{S}$. The linearization of $\mathcal{G}$ to $\mathcal{S}$ as a preprocessing step was first proposed by [16] and has since been applied in, e.g., [18, 44]. In order to construct $\mathcal{S}$, a random-spanning tree $\mathcal{R}$ is picked uniformly at random. A vertex of $\mathcal{R}$ is then chosen, and the graph is fully traversed using a *depth-first search* generating an ordered list $V_{\mathcal{L}} = \left\{i_{l_1}, \ldots, i_{l_{2m+1}}\right\}$ of vertices (where $m \leq |E|$) in the order they were visited. Vertices in V may appear multiple times in $V_{\mathcal{L}}$. A subsequence $V_{\mathcal{L}'} \subseteq V_{\mathcal{L}}$ is then chosen such that each vertex in V appears only once. The line graph $\mathcal{S}$ is then formed by connecting each vertex in $V_{\mathcal{L}'}$ to its immediate neighbors in $V_{\mathcal{L}'}$ with an edge. We denote the edge set of $\mathcal{S}$ by $E_{\mathcal{S}}$ and let $\Phi_{\mathcal{S}}(\boldsymbol{\mu})$ denote the cut-size of the labeling $\boldsymbol{\mu}$ with respect to the linear embedding $\mathcal{S}$.

Surprisingly, as stated in the lemma below, the cut on this linearized graph is no more than twice the cut on the original graph.

Lemma 1 ([16]). *Given a labeling $\boldsymbol{\mu} \in \{-1, 1\}^n$ on a graph $\mathcal{G}$, for the mapping $\mathcal{G} \to \mathcal{R} \to \mathcal{S}$, as above, we have $\Phi_{\mathcal{S}}(\boldsymbol{\mu}) \leq 2\Phi_{\mathcal{R}}(\boldsymbol{\mu}) \leq 2\Phi_{\mathcal{G}}(\boldsymbol{\mu})$.*

By combining the above observations we may reduce the problem of learning on a graph to that of learning on a line graph. In particular, if we have an algorithm with a mistake bound of the form $M \leq \mathcal{O}(\Phi_{\mathcal{G}}(\boldsymbol{\mu}))$ this implies we then may give an *expected* mistake bound of the form $M \leq \mathcal{O}(\Phi^r_{\mathcal{G}}(\boldsymbol{\mu}))$ by first sampling a random spanning tree and then linearizing it as above.

One caveat of this, however, depends on whether `nature` is *oblivious* or *adaptive*. If `nature` is oblivious, we assume that the `learner`'s predictions have no effect on the labels chosen by `nature` (or equivalently all labelings are chosen beforehand). Conversely, if `nature` is adaptive, then `nature`'s labelings are assumed to be adversarially chosen in response to the `learner`'s predictions. In this work we will only state the deterministic mistake bounds in terms of the cut-size, which will hold for oblivious and adaptive adversaries, while the expected bounds in terms of resistance-weighted cut-sizes will hold for an oblivious adversary.

For the remainder of this chapter we let $\Phi^t := \Phi(\boldsymbol{\mu}^t)$, where the cut Φ is either with respect to $\mathcal{G}$ or to the linear embedding $\mathcal{S}$, depending on the algorithm. Finally, we define the max cut of the distinct labelings as $\hat{\Phi} := \max_{k \in \mathcal{K}} \Phi^k$.

3.4 Switching Specialists

In this section we present a new method based on the idea of *specialists*, which we briefly introduced in Section 2.2.1. We will see that although the achieved bounds are slightly worse than other methods for predicting a *single* labeling of a graph, the derived advantage is that it is possible to obtain "competitive" bounds with fast algorithms to predict a sequence of changing graph labelings.

Our inductive bias for this problem will be to predict well when a labeling

has a *small* (resistance-weighted) cut-size. The complementary perspective of this implies that the labeling consists of a *few* uniformly labeled clusters. This suggests maintaining a collection of basis functions where each function is specialized to predict a constant function on a given cluster of vertices. To accomplish this technically, we introduce the following method of applying the idea of *specialists* to graphs. In this context, we introduce a set of specialists, $\mathcal{E}$, where we treat a specialist $\varepsilon \in \mathcal{E}$ as simply a prediction function from an input space to an extended output space with *abstentions*. So for us the input space is just $V = [n]$, the vertices of a graph; and the extended output space is $\{-1, 1, \square\}$ where $\{-1, 1\}$ corresponds to predicted labels of the vertices, but "$\square$" indicates that the specialist abstains from predicting. Thus a specialist *specializes* its prediction to part of the input space, and in our application the specialists correspond to a collection of clusters that cover the graph, each cluster uniformly predicting -1 or 1. We call these functions *cluster specialists*.

In Algorithm 1, SWITCHING CLUSTER SPECIALISTS, we give our switching specialists method. The algorithm maintains a weight vector $\boldsymbol{w}^t \in \Delta_{|\mathcal{E}|}$ over the specialists in which the magnitudes may be interpreted as the current confidence we have in each of the specialists. On each trial we are presented with a node $i_t \in V$, and we define the set of non-abstaining specialists by $\mathcal{A}_t : \{\varepsilon \in \mathcal{E} : \varepsilon(i_t) \neq \square\}$. Given the true label, $y^t \in \{-1, 1\}$ of i_t, we define the set of correct specialists by $\mathcal{Y}_t := \{\varepsilon \in \mathcal{E} : \varepsilon(i_t) = y^t\}$. Our algorithm's updates and analyses are a combination of three standard methods: i) *Halving* loss updates, ii) specialists updates, and iii) (delayed) fixed-share updates. The loss update (3.5) zeros the weight components of incorrectly predicting specialists, while the non-predicting specialists are not updated at all. In (3.5) we give our *delayed* fixed-share style update. Note that our algorithm is also *conservative*, that is, we only update weights on trials that the algorithm makes a mistake. Let $\dot{w}_\varepsilon^t$ be the weight of specialist ε after incorporating the loss incurred on trial t. The standard fixed share update [30] may be written in the following

Algorithm 1 SWITCHING CLUSTER SPECIALISTS

Input: Specialists set $\mathcal{E}$; $\alpha \in [0,1]$

Initialize: $\boldsymbol{w}^1 \leftarrow \frac{1}{|\mathcal{E}|}\mathbf{1}$; $\dot{\boldsymbol{w}}^0 \leftarrow \frac{1}{|\mathcal{E}|}\mathbf{1}$; $\boldsymbol{p} \leftarrow \mathbf{0}$; $m \leftarrow 0$

for $t \leftarrow 1$ to T **do**
  receive $i_t \in V$
  $\mathcal{A}_t \leftarrow \{\varepsilon \in \mathcal{E} : \varepsilon(i_t) \neq \square\}$
  for all $\varepsilon \in \mathcal{A}_t$ **do** ▷ delayed share update
    $w_\varepsilon^t \leftarrow (1-\alpha)^{m-p_\varepsilon}\dot{w}_\varepsilon^{t-1} + \frac{1-(1-\alpha)^{m-p_\varepsilon}}{|\mathcal{E}|}$ (3.5)
  end for
  predict $\hat{y}^t \leftarrow \operatorname{sign}(\sum_{\varepsilon\in\mathcal{A}_t} w_\varepsilon^t\, \varepsilon(i_t))$
  receive $y^t \in \{-1,1\}$
  $\mathcal{Y}_t \leftarrow \{\varepsilon \in \mathcal{E} : \varepsilon(i_t) = y^t\}$
  if $\hat{y}^t \neq y^t$ **then** ▷ loss update

$$\dot{w}_\varepsilon^t \leftarrow \begin{cases} 0 & \varepsilon \in \mathcal{A}_t \cap \bar{\mathcal{Y}}_t \\ \dot{w}_\varepsilon^{t-1} & \varepsilon \notin \mathcal{A}_t \\ w_\varepsilon^t \frac{\boldsymbol{w}^t(\mathcal{A}_t)}{\boldsymbol{w}^t(\mathcal{Y}_t)} & \varepsilon \in \mathcal{Y}_t \end{cases} \tag{3.6}$$

    for all $\varepsilon \in \mathcal{A}_t$ **do**
      $p_\varepsilon \leftarrow m$
    end for
    $m \leftarrow m+1$
  else
    $\dot{\boldsymbol{w}}^t \leftarrow \dot{\boldsymbol{w}}^{t-1}$
  end if
end for

form

$$w_\varepsilon^t = (1-\alpha)\dot{w}_\varepsilon^{t-1} + \frac{\alpha}{|\mathcal{E}|}\,, \tag{3.7}$$

where the parameter $\alpha \in [0,1]$ is interpreted as the switching rate. We revisit this update in more detail in Chapter 4. Although (3.7) superficially appears different to (3.5), in fact these two updates are exactly the same in terms of predictions generated by the algorithm. This is because (3.5) caches updates until the given specialist is again active. The purpose of this computationally is that if the active specialists are, for example, logarithmic in size compared to the total specialist pool, we may then achieve an exponential speedup over (3.7); which in fact we will exploit.

Before we give our mistake bound guarantee for the SWITCHING CLUSTER SPECIALISTS algorithm, we require the following definition of well-formedness

with respect to a comparator set of weights $\boldsymbol{u} \in \Delta_{|\mathcal{E}|}$.

Definition 2 (Well-formedness)**.** *A comparator* $\boldsymbol{u} \in \Delta_{|\mathcal{E}|}$ *is well-formed if* $\forall\ v \in V$*, there exists a* unique $\varepsilon \in \mathcal{E}$ *such that* $\varepsilon(v) \neq \square$ *and* $u_\varepsilon > 0$*, and furthermore there exists a* $\pi \in (0, 1]$ *such that* $\forall \varepsilon \in \mathcal{E} : u_\varepsilon \in \{0, \pi\}$*,*

In other words, a comparator $\boldsymbol{u}$ is said to be well-formed if each specialist in the support of $\boldsymbol{u}$ has the same mass π and these specialists disjointly cover the input space (V). In the next section this definition will allow us to learn a labeling of a graph by learning a set of weights of a suitably chosen specialist set.

We will also require the following defintition of *consistency* of a comparator, $\boldsymbol{u} \in \Delta_{|\mathcal{E}|}$.

Definition 3 (Consistency)**.** *A comparator* $\boldsymbol{u} \in \Delta_{|\mathcal{E}|}$ *is consistent with the labeling* $\boldsymbol{\mu} \in \{-1, 1\}^n$ *if* $\boldsymbol{u}$ *is well-formed and* $u_\varepsilon > 0$ *implies that for all* $v \in V$ *that* $\varepsilon(v) \in \{\mu_v, \square\}$*.*

That is, on every trial there is no active incorrect specialist assigned "mass" by the comparator $\boldsymbol{u}$.

In the following theorem we give our switching specialists bound. We will see that the dominant cost of switching from trial t to $t + 1$ is given by the non-symmetric quantity $J_{\mathcal{E}}(\boldsymbol{u}^t, \boldsymbol{u}^{t+1}) := |\{\varepsilon \in \mathcal{E} : u^t_\varepsilon = 0, u^{t+1}_\varepsilon \neq 0\}|$. That is, we pay only for each new specialist introduced, but we do not pay for "removing" specialists.

Theorem 4. *For a given specialist set,* $\mathcal{E}$*, let* $M_{\mathcal{E}}$ *denote the number of mistakes made in predicting the online sequence* $(i_1, y^1), \ldots, (i_T, y^T)$ *by Algorithm 1. Then,*

$$M_{\mathcal{E}} \leq \frac{1}{\pi^1} \log |\mathcal{E}| + \sum_{t=1}^{T} \frac{1}{\pi^t} \log\left(\frac{1}{1-\alpha}\right) + \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{E}}\big(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}}\big) \log\left(\frac{|\mathcal{E}|}{\alpha}\right), \quad (3.8)$$

for any sequence of ***consistent*** *and* ***well-formed*** *comparators* $\boldsymbol{u}^1, \ldots, \boldsymbol{u}^t \in \Delta_{|\mathcal{E}|}$ *where* $\mathcal{K} := \{k_1 = 1 < \cdots < k_{|\mathcal{K}|}\} := \{t \in [T] : \boldsymbol{u}^t \neq \boldsymbol{u}^{t-1}\} \cup \{1\}$*, and* $\pi^t := \boldsymbol{u}^t(\mathcal{Y}_t)$*.*

Proof. Recall that the cached share update (3.5) is equivalent to performing (3.7). We thus simulate the latter update in our analysis. We first argue the inequality

$$[\![\hat{y}^t \neq y^t]\!] \leq \frac{1}{\boldsymbol{u}^t(\mathcal{Y}_t)} \left(D(\boldsymbol{u}^t, \boldsymbol{w}^t) - D(\boldsymbol{u}^t, \dot{\boldsymbol{w}}^t) \right), \tag{3.9}$$

as this is derived by observing that

$$\begin{aligned} D(\boldsymbol{u}^t, \boldsymbol{w}^t) - D(\boldsymbol{u}^t, \dot{\boldsymbol{w}}^t) &= \sum_{\varepsilon \in \mathcal{E}} u_\varepsilon^t \log \frac{\dot{w}_\varepsilon^t}{w_\varepsilon^t} \\ &= \sum_{\varepsilon \in \mathcal{Y}_t} u_\varepsilon^t \log \frac{\dot{w}_\varepsilon^t}{w_\varepsilon^t} \\ &\geq \boldsymbol{u}^t(\mathcal{Y}_t) [\![\hat{y}^t \neq y^t]\!] , \end{aligned}$$

where the second line follows the fact that $u_\varepsilon^t \log \frac{\dot{w}_\varepsilon^t}{w_\varepsilon^t} = 0$ if $\varepsilon \notin \mathcal{Y}_t$ as either the specialist ε predicts "$\square$" and $\dot{w}_\varepsilon^t = w_\varepsilon^t$ or it predicts incorrectly and hence $u_\varepsilon^t = 0$. The third line follows as for $\varepsilon \in \mathcal{Y}_t$, $\frac{\dot{w}_\varepsilon^t}{w_\varepsilon^t} \geq 2$ if there has been a mistake on trial t and otherwise the ratio is ≥ 1. Indeed, since Algorithm 1 is conservative, this ratio is exactly 1 when no mistake is made on trial t, thus without loss of generality we will assume the algorithm makes a mistake on every trial.

For clarity we will now use simplified notation and let $\pi^t := \boldsymbol{u}^t(\mathcal{Y}_t)$. We now prove the following two inequalities,

$$\frac{1}{\pi^t} \left[D(\boldsymbol{u}^t, \dot{\boldsymbol{w}}^t) - D(\boldsymbol{u}^t, \boldsymbol{w}^{t+1}) \right] \geq -\frac{1}{\pi^t} \log \left(\frac{1}{1-\alpha} \right), \tag{3.10}$$

and

$$\begin{aligned} \frac{1}{\pi^t} D(\boldsymbol{u}^t, \boldsymbol{w}^{t+1}) - \frac{1}{\pi^{t+1}} D(\boldsymbol{u}^{t+1}, \boldsymbol{w}^{t+1}) \geq -\frac{1}{\pi^t} H(\boldsymbol{u}^t) + \frac{1}{\pi^{t+1}} H(\boldsymbol{u}^{t+1}) \\ - J_{\mathcal{E}} \left(\boldsymbol{u}^t, \boldsymbol{u}^{t+1} \right) \log \left(\frac{|\mathcal{E}|}{\alpha} \right). \end{aligned} \tag{3.11}$$

which we will add to (3.9) to create a telescoping sum of relative entropy terms

and entropy terms. For the sake of brevity we will define

$$\tilde{D}^t := \frac{1}{\pi^t} D(\boldsymbol{u}^t, \boldsymbol{w}^{t+1}) - \frac{1}{\pi^{t+1}} D(\boldsymbol{u}^{t+1}, \boldsymbol{w}^{t+1}) \,.$$

Now, (3.10) is proved with the following

$$\begin{aligned} D(\boldsymbol{u}^t, \dot{\boldsymbol{w}}^t) - D(\boldsymbol{u}^t, \boldsymbol{w}^{t+1}) &= \sum_{\varepsilon \in \mathcal{E}} u_\varepsilon^t \log \frac{w_\varepsilon^{t+1}}{\dot{w}_\varepsilon^t} \\ &\geq \sum_{\varepsilon \in \mathcal{E}} u_\varepsilon^t \log \left(\frac{(1-\alpha)\dot{w}_\varepsilon^t}{\dot{w}_\varepsilon^t} \right) \\ &= \log (1-\alpha) \,, \end{aligned}$$

where the inequality has used $w_\varepsilon^{t+1} \geq (1-\alpha)\dot{w}_\varepsilon^t$ from (3.7).

To prove (3.11) we first define the following sets.

$$\begin{aligned} \Theta_t &:= \{\varepsilon \in \mathcal{E} : u_\varepsilon^{t-1} \neq 0, u_\varepsilon^t = 0\} \,, \\ \Psi_t &:= \{\varepsilon \in \mathcal{E} : u_\varepsilon^{t-1} \neq 0, u_\varepsilon^t \neq 0\} \,, \\ \Omega_t &:= \{\varepsilon \in \mathcal{E} : u_\varepsilon^{t-1} = 0, u_\varepsilon^t \neq 0\} \,. \end{aligned}$$

We now expand the following

$$\begin{aligned} \tilde{D}^t &= \frac{1}{\pi^t} D(\boldsymbol{u}^t, \boldsymbol{w}^{t+1}) - \frac{1}{\pi^t} D(\boldsymbol{u}^{t+1}, \boldsymbol{w}^{t+1}) + \frac{1}{\pi^t} D(\boldsymbol{u}^{t+1}, \boldsymbol{w}^{t+1}) \\ &\quad - \frac{1}{\pi^{t+1}} D(\boldsymbol{u}^{t+1}, \boldsymbol{w}^{t+1}) \\ &= \frac{1}{\pi^t} \sum_{\varepsilon \in \mathcal{E}} u_\varepsilon^t \log \left(\frac{u_\varepsilon^t}{w_\varepsilon^{t+1}} \right) - \frac{1}{\pi^t} \sum_{\varepsilon \in \mathcal{E}} u_\varepsilon^{t+1} \log \left(\frac{u_\varepsilon^{t+1}}{w_\varepsilon^{t+1}} \right) \\ &\quad + \frac{1}{\pi^t} \sum_{\varepsilon \in \mathcal{E}} u_\varepsilon^{t+1} \log \left(\frac{u_\varepsilon^{t+1}}{w_\varepsilon^{t+1}} \right) - \frac{1}{\pi^{t+1}} \sum_{\varepsilon \in \mathcal{E}} u_\varepsilon^{t+1} \log \left(\frac{u_\varepsilon^{t+1}}{w_\varepsilon^{t+1}} \right) \\ &= -\frac{1}{\pi^t} H(\boldsymbol{u}^t) + \frac{1}{\pi^t} H(\boldsymbol{u}^{t+1}) + \sum_{\varepsilon \in \mathcal{E}} \left(\frac{u_\varepsilon^t}{\pi^t} - \frac{u_\varepsilon^{t+1}}{\pi^t} \right) \log \left(\frac{1}{w_\varepsilon^{t+1}} \right) \\ &\quad - \frac{1}{\pi^t} H(\boldsymbol{u}^{t+1}) + \frac{1}{\pi^{t+1}} H(\boldsymbol{u}^{t+1}) \\ &\quad + \sum_{\varepsilon \in \mathcal{E}} \left(\frac{u_\varepsilon^{t+1}}{\pi^t} - \frac{u_\varepsilon^{t+1}}{\pi^{t+1}} \right) \log \left(\frac{1}{w_\varepsilon^{t+1}} \right) . \end{aligned} \tag{3.12}$$

Recall that a comparator $\boldsymbol{u} \in \Delta_{|\mathcal{E}|}$ is *well-formed* if $\forall\ v \in V$, there exists a *unique* $\varepsilon \in \mathcal{E}$ such that $\varepsilon(v) \neq \square$ and $u_\varepsilon > 0$, and furthermore there exists a $\pi \in (0, 1]$ such that $\forall \varepsilon \in \mathcal{E} : u_\varepsilon \in \{0, \pi\}$, i.e., each specialist in the support of $\boldsymbol{u}$ has the same mass π and these specialists disjointly cover the input space (V). Thus, by collecting terms into the three sets Θ_{t+1}, Ψ_{t+1}, and Ω_{t+1} we have

$$\begin{aligned}
\sum_{\varepsilon\in\mathcal{E}} \left(\frac{u_\varepsilon^t}{\pi^t} - \frac{u_\varepsilon^{t+1}}{\pi^t}\right) \log\left(\frac{1}{w_\varepsilon^{t+1}}\right) &= \sum_{\varepsilon\in\Theta_{t+1}} \frac{u_\varepsilon^t}{\pi^t} \log\left(\frac{1}{w_\varepsilon^{t+1}}\right) \\
&\quad + \sum_{\varepsilon\in\Psi_{t+1}} \left(\frac{u_\varepsilon^t}{\pi^t} - \frac{u_\varepsilon^{t+1}}{\pi^t}\right) \log\left(\frac{1}{w_\varepsilon^{t+1}}\right) \\
&\quad - \sum_{\varepsilon\in\Omega_{t+1}} \frac{u_\varepsilon^{t+1}}{\pi^t} \log\left(\frac{1}{w_\varepsilon^{t+1}}\right) \\
&= \sum_{\varepsilon\in\Theta_{t+1}} \frac{u_\varepsilon^t}{\pi^t} \log\left(\frac{1}{w_\varepsilon^{t+1}}\right) \\
&\quad + \sum_{\varepsilon\in\Psi_{t+1}} \left(1 - \frac{u_\varepsilon^{t+1}}{\pi^t}\right) \log\left(\frac{1}{w_\varepsilon^{t+1}}\right) \\
&\quad - \sum_{\varepsilon\in\Omega_{t+1}} \frac{u_\varepsilon^{t+1}}{\pi^t} \log\left(\frac{1}{w_\varepsilon^{t+1}}\right), \qquad (3.13)
\end{aligned}$$

and similarly

$$\begin{aligned}
\sum_{\varepsilon\in\mathcal{E}} \left(\frac{u_\varepsilon^{t+1}}{\pi^t} - \frac{u_\varepsilon^{t+1}}{\pi^{t+1}}\right) \log\left(\frac{1}{w_\varepsilon^{t+1}}\right) &= \sum_{\varepsilon\in\Psi_{t+1}} \left(\frac{u_\varepsilon^{t+1}}{\pi^t} - 1\right) \log\left(\frac{1}{w_\varepsilon^{t+1}}\right) \\
&\quad + \sum_{\varepsilon\in\Omega_{t+1}} \left(\frac{u_\varepsilon^{t+1}}{\pi^t} - 1\right) \log\left(\frac{1}{w_\varepsilon^{t+1}}\right). \qquad (3.14)
\end{aligned}$$

Substituting (3.13) and (3.14) into (3.12) and simplifying gives

$$\begin{aligned}
\tilde{D}^t &= -\frac{1}{\pi^t} H(\boldsymbol{u}^t) + \frac{1}{\pi^{t+1}} H(\boldsymbol{u}^{t+1}) + \sum_{\varepsilon\in\Theta_{t+1}} \frac{u_\varepsilon^t}{\pi^t} \log\left(\frac{1}{w_\varepsilon^{t+1}}\right) - \sum_{\varepsilon\in\Omega_{t+1}} \log\left(\frac{1}{w_\varepsilon^{t+1}}\right) \\
&\geq -\frac{1}{\pi^t} H(\boldsymbol{u}^t) + \frac{1}{\pi^{t+1}} H(\boldsymbol{u}^{t+1}) - |\Omega_{t+1}| \log\left(\frac{|\mathcal{E}|}{\alpha}\right), \qquad (3.15)
\end{aligned}$$

where the inequality has used the fact that $\frac{\alpha}{|\mathcal{E}|} \leq w_\varepsilon^{t+1} \leq 1$ from (3.7).

Summing over all trials then leaves a telescoping sum of relative entropy

terms, a cost of $\frac{1}{\pi^t}\log\left(\frac{1}{1-\alpha}\right)$ on each trial, and $|\Omega_{t+1}|\log\left(\frac{|\mathcal{E}|}{\alpha}\right)$ for each switch. Thus,

$$\sum_{t=1}^{T}[\![\hat{y}^t \neq y^t]\!] \leq \frac{1}{\pi^1}D(\boldsymbol{u}^1,\boldsymbol{w}^1) + \frac{1}{\pi^1}H(\boldsymbol{u}^1) + \sum_{t=1}^{T}\frac{1}{\pi^t}\log\left(\frac{1}{1-\alpha}\right) + \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{E}}\left(\boldsymbol{u}^{k_i},\boldsymbol{u}^{k_{i+1}}\right)\log\left(\frac{|\mathcal{E}|}{\alpha}\right), \tag{3.16}$$

where $J_{\mathcal{E}}(\boldsymbol{u}^{k_i},\boldsymbol{u}^{k_{i+1}}) = |\Omega_{k_{i+1}}|$, and since $\boldsymbol{w}^1 = \frac{1}{|\mathcal{E}|}\mathbf{1}$, we can combine the remaining entropy and relative entropy terms to give $\frac{1}{\pi^1}D(\boldsymbol{u}^1,\boldsymbol{w}^1)+\frac{1}{\pi^1}H(\boldsymbol{u}^1) = \frac{1}{\pi^1}\log|\mathcal{E}|$, concluding the proof. □

The bound in the above theorem depends crucially on the best sequence of *consistent* and *well-formed* comparators $\boldsymbol{u}^1,\ldots,\boldsymbol{u}^T$. The consistency requirement implies that on every trial there is no active incorrect specialist assigned "mass" ($\boldsymbol{u}^t(\mathcal{A}_t \setminus \mathcal{Y}_t) = 0$). We may eliminate the consistency requirement by "softening" the loss update (3.5). The well-formedness requirement may also be eliminated at considerable complication to the form of the bound.

Note that the above bound is "smooth" in that it scales with a gradual change in the comparator. In the next section we describe the novel specialists sets that we have tailored to graph-label prediction so that a small change in comparator corresponds to a small change in a graph labeling.

3.5 Cluster Specialists

In order to construct the *cluster specialists* over a graph $\mathcal{G} = (V, E)$, we first construct a line graph as described in Section 3.3. A cluster specialist is then defined by $\varepsilon_y^{l,r}(\cdot)$ which maps $V \to \{-1, 1, \square\}$ where

$$\varepsilon_y^{l,r}(v) := \begin{cases} y & l \leq v \leq r\,, \\ \square & \text{otherwise.} \end{cases}$$

Hence cluster specialist $\varepsilon_y^{l,r}(v)$ corresponds to a function that predicts the label y if vertex v lies between vertices l and r and abstains otherwise. Recall that by sampling a random spanning tree, the expected cut-size of a labeling on the spine is no more than twice the resistance-weighted cut-size on $\mathcal{G}$. Thus, given a labeled graph with a small resistance-weighted cut-size with densely interconnected clusters and modest intra-cluster connections, this implies a cut-bracketed linear segment on the spine will, in expectation, roughly correspond to one of the original dense clusters.

We now introduce two sets of cluster specialists for our problem of predicting on a graph. Before we do so, however, we require the following definitions which will characterize these sets.

Definition 5 (Covering)**.** *We say that a subset of specialists $\mathcal{C}_{\boldsymbol{\mu}} \subseteq \mathcal{E} \subseteq 2^{\{-1,1,\square\}^n}$ from basis $\mathcal{E}$* covers *a labeling $\boldsymbol{\mu} \in \{-1,1\}^n$ if for all $v \in V = [n]$ and $\varepsilon \in \mathcal{C}_{\boldsymbol{\mu}}$ we have $\varepsilon(v) \in \{\mu_v, \square\}$ and if $v \in V$ then there exists $\varepsilon \in \mathcal{C}_{\boldsymbol{\mu}}$ such that $\varepsilon(v) = \mu_v$.*

Definition 6 (Completeness)**.** *The basis $\mathcal{E} \subseteq 2^{\{-1,1,\square\}^n}$ is* complete *if every labeling $\boldsymbol{\mu} \in \{-1,1\}^n$ is covered by some $\mathcal{C}_{\boldsymbol{\mu}} \subseteq \mathcal{E}$.*

With these definitions, we now consider two complete basis sets of cluster specialists.

3.5.1 Basis $\mathcal{F}_n$

We first introduce the basis set

$$\mathcal{F}_n := \{\varepsilon_y^{l,r} : l, r \in [n], l \leq r; y \in \{-1,1\}\}.$$

The basis $\mathcal{F}_n$ is clearly complete, and in fact has the following approximation property: for any $\boldsymbol{\mu} \in \{-1,1\}^n$ there exists a covering set $\mathcal{C}_{\boldsymbol{\mu}} \subseteq \mathcal{F}_n$ such that $|\mathcal{C}_{\boldsymbol{\mu}}| = \Phi_{\mathcal{S}}(\boldsymbol{\mu}) + 1$. This follows directly as a line with $k-1$ cuts is divided into k segments. See Figure 3.2 and Figure 3.3 for a visualization of the basis $\mathcal{F}_n$ for $n = 8$.

We now illustrate the use of basis $\mathcal{F}_n$ to predict the labeling of a graph. For simplicity, we illustrate by considering the problem of predicting a single

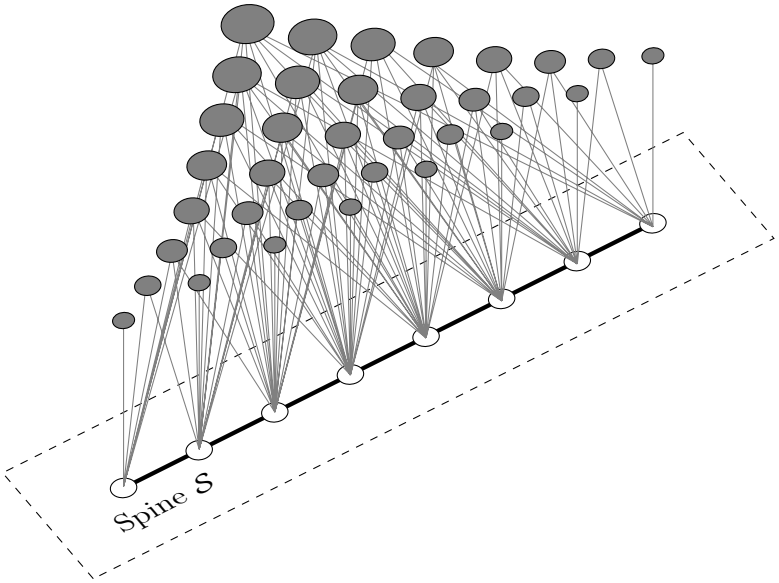


Figure 3.2: Representation of basis set $\mathcal{F}_{1,8}$ for a single parameter y. Gray circles represent specialists. A line connecting specialist $\varepsilon_y^{l,r}$ to node $v \in \mathcal{S}$ implies $\varepsilon_y^{l,r}(v) \neq \square$.

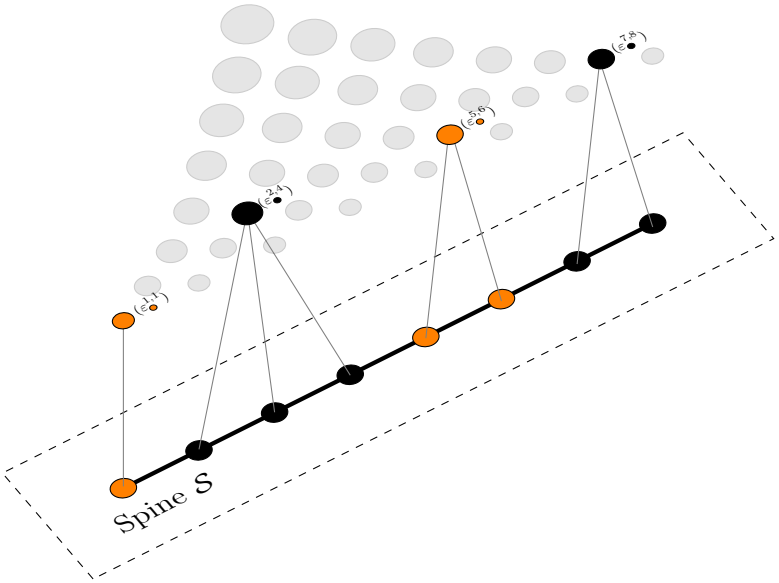


Figure 3.3: Representation of the smallest subset of basis set $\mathcal{F}_{1,8}$ needed to cover a given labeling $\boldsymbol{\mu}$ of cut-size $\Phi_{\mathcal{S}}(\boldsymbol{\mu}) = 3$.

graph labeling without switching. As there is no switch we will set $\alpha := 0$ and thus if the graph is labeled with $\boldsymbol{\mu} \in \{-1, 1\}^n$ with cut-size $\Phi_{\mathcal{S}}(\boldsymbol{\mu})$ then we will need $\Phi_{\mathcal{S}}(\boldsymbol{\mu})+1$ specialists to predict the labeling and thus the comparators may be post-hoc optimally determined so that $\boldsymbol{u} = \boldsymbol{u}^1 = \cdots = \boldsymbol{u}^T$ and there will be $\Phi_{\mathcal{S}}(\boldsymbol{\mu}) + 1$ components of $\boldsymbol{u}$ each with "weight" $\frac{1}{\Phi_{\mathcal{S}}(\boldsymbol{\mu})+1}$, thus $\frac{1}{\pi^1} = \Phi_{\mathcal{S}}(\boldsymbol{\mu}) + 1$, since there will be only one specialist (with non-zero weight) active per trial. Since the cardinality of $\mathcal{F}_n$ is $n^2 + n$, by substituting into (3.8) we have that the number of mistakes will be bounded by $(\Phi_{\mathcal{S}}(\boldsymbol{\mu}) + 1) \log (n^2 + n)$. Note for a single graph labeling on a spine this bound is not much worse than the best known result [16, Theorem 4]. In terms of computation time, however, it is significantly slower than the algorithm in [16] requiring $\Theta(n^2)$ time to predict on a typical trial since on average there are $\Theta(n^2)$ specialists active per trial.

3.5.2 Basis $\mathcal{B}_{1,n}$

We now introduce the basis $\mathcal{B}_{1,n}$ which has $\Theta(n)$ specialists and only requires $\mathcal{O}(\log n)$ time per trial to predict with only a small increase in bound. The basis is defined as

$$\mathcal{B}_{p,q} := \begin{cases} \{\varepsilon_{-1}^{p,q}, \varepsilon_1^{p,q}\} & p = q \\ \{\varepsilon_{-1}^{p,q}, \varepsilon_1^{p,q}\} \cup \mathcal{B}_{p,\lfloor \frac{p+q}{2} \rfloor} \cup \mathcal{B}_{\lfloor \frac{p+q}{2} \rfloor + 1,q} & p \neq q \,, \end{cases}$$

and is analogous to a binary tree. See Figure 3.4 and Figure 3.5 for a visualization of the basis $\mathcal{B}_{1,n}$ for $n = 8$. We have the following approximation property for $\mathcal{B}_n := \mathcal{B}_{1,n}$,

Proposition 7. *The basis $\mathcal{B}_n$ is complete. Furthermore, for any labeling $\boldsymbol{\mu} \in \{-1, 1\}^n$ there exists a covering set $\mathcal{C}_{\boldsymbol{\mu}} \subseteq \mathcal{B}_n$ such that $|\mathcal{C}_{\boldsymbol{\mu}}| \leq 2(\Phi_{\mathcal{S}}(\boldsymbol{\mu}) + 1)\lceil \log_2 \frac{n}{2} \rceil$ for $n > 2$.*

Proof. We first give a brief intuition of the proof; any required terms will be defined more completely later. For a given labeling $\boldsymbol{\mu} \in \{-1, 1\}^n$ of cut-size $\Phi_{\mathcal{S}}(\boldsymbol{\mu})$, the spine $\mathcal{S}$ can be cut into $\Phi_{\mathcal{S}}(\boldsymbol{\mu}) + 1$ *clusters*, where a cluster is a contiguous segment of vertices with the same label. We will upper bound

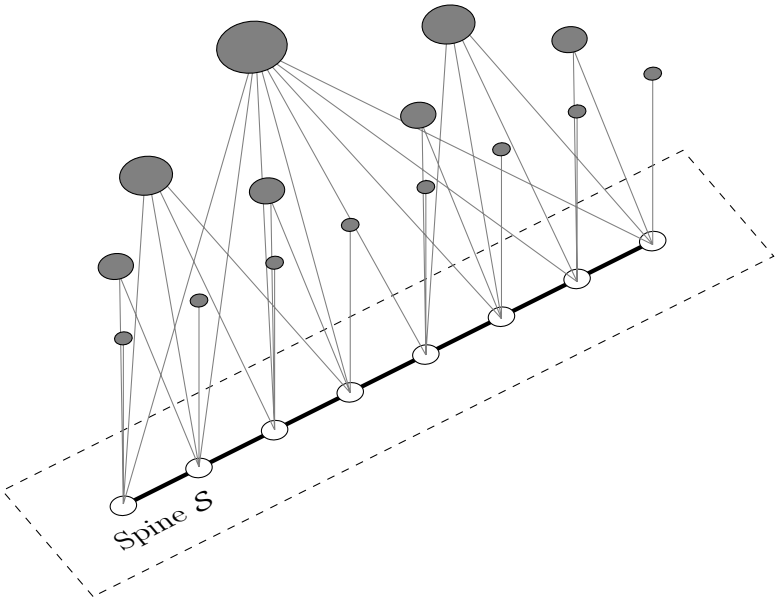


Figure 3.4: Representation of basis set $\mathcal{B}_{1,8}$ for a single parameter y. Gray circles represent specialists. A line connecting specialist $\varepsilon_y^{l,r}$ to node $v \in \mathcal{S}$ implies $\varepsilon_y^{l,r}(v) \neq \square$.

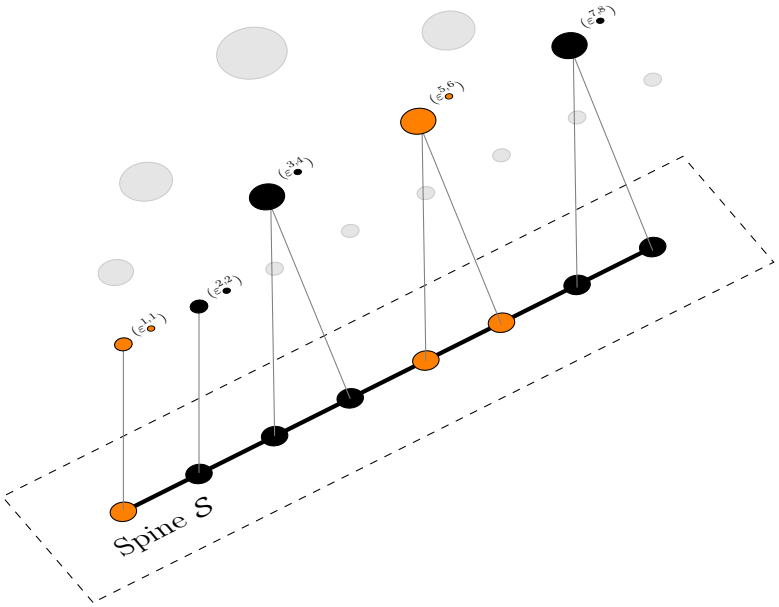


Figure 3.5: Representation of the smallest subset of basis set $\mathcal{B}_{1,8}$ needed to cover a given labeling $\boldsymbol{\mu}$ of cut-size $\Phi_{\mathcal{S}}(\boldsymbol{\mu}) = 3$.

the maximum number of cluster specialists required to cover a single cluster, and therefore obtain an upper bound for $|\mathcal{C}_{\boldsymbol{\mu}}|$ by summing over the $\Phi_{\mathcal{S}}(\boldsymbol{\mu}) + 1$ clusters.

Without loss of generality, we assume $n = 2^r$ for some integer r and thus the structure of $\mathcal{B}_n$ is analogous to a perfect binary tree of depth $d = \log_2 n$. Indeed, for a fixed label parameter y we will adopt the terminology of binary trees such that for instance we say specialist $\varepsilon_y^{i,j}$ for $i \neq j$ has a so-called *left-child* $\varepsilon_y^{i,\lfloor \frac{i+j}{2} \rfloor}$ and *right-child* $\varepsilon_y^{\lceil \frac{i+j}{2} \rceil, j}$. Similarly, we say that $\varepsilon_y^{i,\lfloor \frac{i+j}{2} \rfloor}$ and $\varepsilon_y^{\lceil \frac{i+j}{2} \rceil, j}$ are *siblings*, and $\varepsilon_y^{i,j}$ is their *parent*. Note that any specialist is both an ancestor and a descendant of itself, and a proper descendant of a specialist is a descendant of one of its children. Finally the *depth* of specialist $\varepsilon_y^{i,j}$ is defined to be equal to the depth of the corresponding node in a binary tree, such that $\varepsilon_y^{1,n}$ is of depth 0, $\varepsilon_y^{1,\frac{n}{2}}$ and $\varepsilon_y^{\frac{n}{2}+1,n}$ are of depth 1, etc.

The first claim of the proposition is easy to prove as $\{\varepsilon_{-1}^{i,i}, \varepsilon_1^{i,i} : i \in [n]\} \subset \mathcal{B}_n$ and thus any labeling $\boldsymbol{\mu} \in \{-1, 1\}^n$ can be covered. We now prove the second claim of the proposition.

We will denote a uniformly-labeled contiguous segment of vertices by the pair (l, r), where $l, r \in [n]$ are the two end vertices of the segment. For completeness, we will allow the trivial case when $l = r$. Given a labeling $\boldsymbol{\mu} \in \{-1, 1\}^n$, let $\mathcal{L}_{\boldsymbol{\mu}} := \{(l, r) : 1 \leq l \leq r \leq n; \mu_l = \ldots = \mu_r; \mu_{l-1} \neq \mu_l; \mu_{r+1} \neq \mu_r\}$ be the set of maximum-sized contiguous segments of uniformly-labeled vertices. Note that μ_{l-1} or μ_{r+1} may be vacuous. When the context is clear, we will also describe (l, r) as a *cluster*, and as the set of vertices $\{l, \ldots, r\}$.

For a given $\boldsymbol{\mu} \in \{-1, 1\}^n$ and cluster $(l, r) \in \mathcal{L}_{\boldsymbol{\mu}}$, we say $\mathcal{B}_{(l,r)} \subseteq \mathcal{B}_n$ is an (l, r)-covering set with respect to $\boldsymbol{\mu}$ if for all $\varepsilon_y^{i,j} \in \mathcal{B}_{(l,r)}$ we have $l \leq i, j \leq r$, and if for all $k \in (l, r)$ there exists some $\varepsilon_y^{i,j} \in \mathcal{B}_{(l,r)}$ such that $i \leq k \leq j$ and $y = \mu_k$. That is, every vertex in the cluster is "covered" by at least one specialist and no specialists cover any vertices $k \notin (l, r)$. We define $\Gamma^{(l,r)}$ to be the set of all possible (l, r)-covering sets with respect to $\boldsymbol{\mu}$.

We now define

$$\delta(\mathcal{B}_{(l,r)}) := |\mathcal{B}_{(l,r)}|$$

to be the *complexity* of $\mathcal{B}_{(l,r)} \in \Gamma^{(l,r)}$.

For a given $\boldsymbol{\mu} \in \{-1,1\}^n$ and cluster $(l,r) \in \mathcal{L}_{\boldsymbol{\mu}}$, we wish to produce an (l,r)-covering set of *minimum* complexity, which we denote $\mathcal{B}^*_{(l,r)} := \underset{\mathcal{B}_{(l,r)} \in \Gamma^{(l,r)}}{\arg\min}\ \delta(\mathcal{B}_{(l,r)})$. Note that an (l,r)-covering set of minimum complexity cannot contain any two specialists which are siblings since they can be removed from the set and replaced by their parent specialist.

Lemma 8. *For any $\boldsymbol{\mu} \in \{-1,1\}^n$, for any $(l,r) \in \mathcal{L}_{\boldsymbol{\mu}}$, the (l,r)-covering set of minimum complexity, $\mathcal{B}^*_{(l,r)} = \underset{\mathcal{B}_{(l,r)} \in \Gamma^{(l,r)}}{\arg\min}\ \delta(\mathcal{B}_{(l,r)})$ contains at most two specialists of each unique depth.*

Proof. We first give an intuitive sketch of the proof. For a given $\boldsymbol{\mu} \in \{-1,1\}^n$ and cluster $(l,r) \in \mathcal{L}_{\boldsymbol{\mu}}$ assume that there are at least three specialists of equal depth in $\mathcal{B}^*_{(l,r)}$, then any of these specialists that are in the "middle" may be removed, along with any of their siblings or proper descendants that are also members of $\mathcal{B}^*_{(l,r)}$ without creating any "holes" in the covering, decreasing the complexity of $\mathcal{B}^*_{(l,r)}$.

We use a proof by contradiction. Suppose for contradiction that for a given $\boldsymbol{\mu} \in \{-1,1\}^n$ and $(l,r) \in \mathcal{L}_{\boldsymbol{\mu}}$, the (l,r)-covering set of minimum complexity, $\mathcal{B}^*_{(l,r)}$, contains three distinct specialists of the same depth, $\varepsilon_y^{a,b}, \varepsilon_y^{c,d}, \varepsilon_y^{e,f}$. Without loss of generality, let $a,b < c,d < e,f$. Note that we have $l \le a < f \le r$. We consider the following two possible scenarios: when two of the three specialists are siblings, and when none are.

If $\varepsilon_y^{a,b}$ and $\varepsilon_y^{c,d}$ are siblings, then we have $\varepsilon_y^{a,d} \in \mathcal{B}_n$ and thus $\{\varepsilon_y^{a,d}\} \cup \mathcal{B}^*_{(l,r)} \setminus \{\varepsilon_y^{a,b}, \varepsilon_y^{c,d}\}$ is an (l,r)-covering set of smaller complexity, leading to a contradiction. The equivalent argument holds if $\varepsilon_y^{c,d}$ and $\varepsilon_y^{e,f}$ are siblings.

If none are siblings, then let $\varepsilon_y^{c',d'}$ be the sibling of $\varepsilon_y^{c,d}$ and let $\varepsilon_y^{C,D}$ be the parent of $\varepsilon_y^{c,d}$ and $\varepsilon_y^{c',d'}$. Note that $a,b < c',d',c,d$ and $c',d',c,d < e,f$ and hence $l < C < D < r$. If an ancestor of $\varepsilon_y^{C,D}$ is in $\mathcal{B}^*_{(l,r)}$, then $\mathcal{B}^*_{(l,r)} \setminus \{\varepsilon_y^{c,d}\}$ is an

(l, r)-covering set of smaller complexity, leading to a contradiction. Alternatively, if no ancestor of $\varepsilon_y^{C,D}$ is in $\mathcal{B}^*_{(l,r)}$, then $\varepsilon_y^{c',d'}$ or some of its proper descendants must be in $\mathcal{B}^*_{(l,r)}$, otherwise there exists some vertex $k' \in (c', d')$ such that there exists no specialist $\varepsilon_y^{i,j} \in \mathcal{B}^*_{(l,r)}$ such that $i \leq k' \leq j$, and therefore $\mathcal{B}^*_{(l,r)}$ would not be an (l, r)-covering set. Let $\varepsilon_y^{p,q}$ be a descendant of $\varepsilon_y^{c',d'}$ which is contained in $\mathcal{B}^*_{(l,r)}$. Then $\{\varepsilon_y^{C,D}\} \cup \mathcal{B}^*_{(l,r)} \setminus \{\varepsilon_y^{c,d}, \varepsilon_y^{p,q}\}$ is an (l, r)-covering set of smaller complexity, leading to a contradiction.

We conclude that there can be no more than 2 specialists of the same depth in $\mathcal{B}^*_{(l,r)}$ for any $\boldsymbol{\mu} \in \{-1, 1\}^n$ and any $(l, r) \in \mathcal{L}_{\boldsymbol{\mu}}$. □

We now prove an upper bound on the maximum minimum-complexity of an (l, r)-covering set under any labeling $\boldsymbol{\mu}$.

Corollary 9. *For all* $\boldsymbol{\mu} \in \{-1, 1\}^n$,

$$\max_{(l,r)\in\mathcal{L}_{\boldsymbol{\mu}}} \min_{\mathcal{B}_{(l,r)}\in\Gamma^{(l,r)}} \delta(\mathcal{B}_{(l,r)}) \leq 2 \log \frac{n}{2} \,. \tag{3.17}$$

Proof. For any $\boldsymbol{\mu} \in \{-1, 1\}^n$, and $(l, r) \in \mathcal{L}_{\boldsymbol{\mu}}$, since $\mathcal{B}^*_{(l,r)}$ can contain at most 2 specialists of the same depth (Lemma 8) an (l, r)-covering set of minimum-complexity can have at most two specialists of depths $2, 3, \ldots, d$. This set cannot contain two specialists of depth 1 as they are siblings. This upper bounds the maximum minimum-complexity of any (l, r)-covering set by $2(d - 2) = 2 \log \frac{n}{2}$. □

Finally we conclude that for any labeling $\boldsymbol{\mu} \in \{-1, 1\}^n$ of cut-size $\Phi_{\mathcal{S}}(\boldsymbol{\mu})$, there exists $\mathcal{C}_{\boldsymbol{\mu}} \subseteq \mathcal{B}_n$ such that $|\mathcal{C}_{\boldsymbol{\mu}}| \leq 2 \log_2 (\frac{n}{2})(\Phi_{\mathcal{S}}(\boldsymbol{\mu}) + 1)$. □

From a computational perspective, the binary tree structure ensures that there are only $\Theta(\log n)$ specialists active per trial, leading to an exponential speed-up in prediction. A similar set of specialists were used for obtaining adaptive-regret bounds in [36, 39] and data-compression in [45]. In those works, however, the "binary tree" structure is over the time dimension (trial sequence), whereas in this work the binary tree is over the space dimension (graph), and a

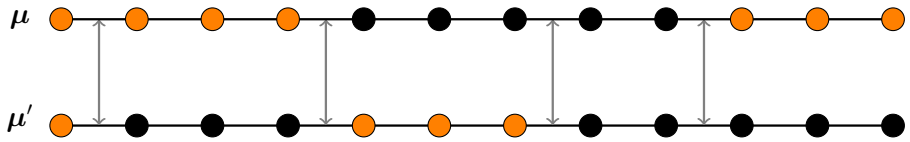


Figure 3.6: An example of two labelings, $\boldsymbol{\mu}$ and $\boldsymbol{\mu}'$, on $\mathcal{S}$. In this case the Hamming-like divergence on cut edges $H(\boldsymbol{\mu}, \boldsymbol{\mu}') = 4$. Observe that $H(\boldsymbol{\mu}, \boldsymbol{\mu}') \leq 2\|\boldsymbol{\mu} - \boldsymbol{\mu}'\|_1$ and is often significantly smaller.

fixed-share update is used to obtain adaptivity over the time dimension.[2] We also note that in [46] the Fixed Share update was combined with specialists and applied to real-world data. In their setting however, it is assumed that the "good" specialist is always active during a given segment, and thus not applicable to our setting of learning disjoint specialists which cover the input space, and can therefore be inactive during these segments.

In the corollary that follows we will exploit the fact that by making the algorithm *conservative* we may reduce the usual $\log T$ term in the mistake bound induced by a fixed-share update to $\log \log T$. A conservative algorithm only updates the specialists' weights on trials on which a mistake is made. Furthermore, the bound given in the following corollary is *smooth* as the cost per switch will be measured with a Hamming-like divergence H on the "cut" edges between successive labelings, defined as

$$H(\boldsymbol{\mu}, \boldsymbol{\mu}') := \sum_{(i,j) \in E_{\mathcal{S}}} [\![\, [\![[\![\mu_i \neq \mu_j]\!] \vee [\![\mu'_i \neq \mu'_j]\!]]\!] \ \wedge \ [\![[\![\mu_i \neq \mu'_i]\!] \vee [\![\mu_j \neq \mu'_j]\!]]\!] \,]\!] \, .$$

See Figure 3.6 for an illustration of this divergence between labelings.

Observe that $H(\boldsymbol{\mu}, \boldsymbol{\mu}')$ is smaller than twice the hamming distance between $\boldsymbol{\mu}$ and $\boldsymbol{\mu}'$ and is often significantly smaller. To achieve the bounds we will now seek to upper bound the divergence J in terms of H.

First, recall the definition of consistency of a comparator given in Definition 3 (that $\boldsymbol{u}(\mathcal{A}_t \setminus \mathcal{Y}_t) = 0$ for all $t \in [T]$). A subtlety is that there are many distinct sets of specialists consistent with a given labeling. For example,

[2] An interesting open problem is to try to find good bounds and time complexity with sets of specialists over *both* the time and space dimensions.

consider a uniform labeling on $\mathcal{S}$. One may "cover" this labeling with a single specialist or alternatively n specialists, one covering each vertex. For the sake of simplicity in bounds we will always choose the smallest set of covering specialists. Thus we introduce the following notion of *minimal-consistency*.

Definition 10 (Minimal consistency)**.** *A comparator $\boldsymbol{u} \in \Delta_{|\mathcal{E}|}$ is minimal-consistent with the labeling $\boldsymbol{\mu} \in \{-1, 1\}^n$ if it is consistent with $\boldsymbol{\mu}$ and the cardinality of its support set $|\{u_\varepsilon : u_\varepsilon > 0\}|$ is the minimum of all comparators consistent with $\boldsymbol{\mu}$.*

We now have the following proposition which upper bounds the divergence J in terms of H.

Proposition 11. *For a linearized graph $\mathcal{S}$, for comparators $\boldsymbol{u}, \boldsymbol{u}' \in \Delta_{|\mathcal{F}_n|}$ that are minimal-consistent with $\boldsymbol{\mu}$ and $\boldsymbol{\mu}'$ respectively,*

$$J_{\mathcal{F}_n}(\boldsymbol{u}, \boldsymbol{u}') \leq \min\left(2H(\boldsymbol{\mu}, \boldsymbol{\mu}'), \Phi_{\mathcal{S}}(\boldsymbol{\mu}') + 1\right).$$

Proof. We will prove both inequalities separately. We first show that $J_{\mathcal{F}_n}(\boldsymbol{u}, \boldsymbol{u}') \leq \Phi_{\mathcal{S}}(\boldsymbol{\mu}') + 1$. This follows directly from the fact that $J_{\mathcal{E}}(\boldsymbol{u}, \boldsymbol{u}') := |\{\varepsilon \in \mathcal{E} : u_\varepsilon = 0, u'_\varepsilon \neq 0\}|$ and therefore

$$J_{\mathcal{F}_n}(\boldsymbol{u}, \boldsymbol{u}') \leq |\{\varepsilon \in \mathcal{F}_n : u'_\varepsilon \neq 0\}| = \Phi_{\mathcal{S}}(\boldsymbol{\mu}') + 1\,.$$

We now prove $J_{\mathcal{F}_n}(\boldsymbol{u}, \boldsymbol{u}') \leq 2H(\boldsymbol{\mu}, \boldsymbol{\mu}')$. Recall that if $\boldsymbol{\mu} \neq \boldsymbol{\mu}'$ then by definition of the minimal-consistent comparators $\boldsymbol{u}$ and $\boldsymbol{u}'$, the set $\{\varepsilon \in \mathcal{F}_n : u_\varepsilon = 0, u'_\varepsilon \neq 0\}$ corresponds to the set of maximum-sized contiguous segments of vertices in $\mathcal{S}$ sharing the same label in the labeling $\boldsymbol{\mu}'$ that did not exist in the labeling $\boldsymbol{\mu}$. From here on we refer to a maximum-sized contiguous segment as just a contiguous segment.

When switching from labeling $\boldsymbol{\mu}$ to $\boldsymbol{\mu}'$, we consider the following three cases. First, when a non-cut edge (with respect to $\boldsymbol{\mu}$) becomes a cut edge (with respect to $\boldsymbol{\mu}'$), second when a cut edge (with respect to $\boldsymbol{\mu}$) becomes a non-cut

edge (with respect to $\boldsymbol{\mu}'$), and lastly when a cut edge remains a cut edge, but the labelings of the two corresponding vertices are "swapped."

Case 1: For an edge $(i,j) \in E_{\mathcal{S}}$ such that $[\![\mu_i = \mu_j]\!] \wedge [\![\mu'_i \neq \mu'_j]\!]$ there exists two new contiguous segments of vertices sharing the same label that did not exist in the labeling $\boldsymbol{\mu}$, their boundary being the edge (i,j).

Case 2: Conversely for an edge $(i,j) \in E_{\mathcal{S}}$ such that $[\![\mu_i \neq \mu_j]\!] \wedge [\![\mu'_i = \mu'_j]\!]$ there exists one new contiguous segment of vertices sharing the same label that did not exist in the labeling $\boldsymbol{\mu}$, that segment will contain the edge (i,j).

Case 3: Finally for an edge $(i,j) \in E_{\mathcal{S}}$ such that

$$[\![[\![\mu_i \neq \mu_j]\!] \wedge [\![\mu'_i \neq \mu'_j]\!]]\!] \wedge [\![[\![\mu_i \neq \mu'_i]\!] \wedge [\![\mu_j \neq \mu'_j]\!]]\!]$$

there exists two new contiguous segments of vertices sharing the same label that did not exist in the labeling $\boldsymbol{\mu}$, their boundary being the edge (i,j).

We conclude that the number of new contiguous segments of vertices sharing the same label that did not exist in the labeling $\boldsymbol{\mu}$ is upper bounded by

$$2 \sum_{(i,j)\in E_{\mathcal{S}}} [\![[\![\mu_i \neq \mu_j]\!] \vee [\![\mu'_i \neq \mu'_j]\!]]\!] \wedge [\![[\![\mu_i \neq \mu'_i]\!] \vee [\![\mu_j \neq \mu'_j]\!]]\!] .$$

□

In the following corollary we summarize the results of the SCS algorithm using the basis sets $\mathcal{F}_n$ and $\mathcal{B}_n$ with an optimally-tuned switching parameter α.

Corollary 12. *For a connected n-vertex graph $\mathcal{G}$ and with randomly sampled spine $\mathcal{S}$, the number of mistakes made in predicting the online sequence $(i_1, y^1), \ldots, (i_T, y^T)$ by the SCS algorithm with optimally-tuned α is upper bounded with basis $\mathcal{F}_n$ by*

$$\mathcal{O}\left(\Phi^1 \log n + \sum_{i=1}^{|\mathcal{K}|-1} H(\boldsymbol{\mu}^{k_i}, \boldsymbol{\mu}^{k_{i+1}}) \left(\log n + \log|\mathcal{K}| + \log\log T\right)\right)$$

and with basis $\mathcal{B}_n$ *by*

$$\mathcal{O}\left(\left(\Phi^1 \log n + \sum_{i=i}^{|\mathcal{K}|-1} H(\boldsymbol{\mu}^{k_i}, \boldsymbol{\mu}^{k_{i+1}}) \left(\log n + \log|\mathcal{K}| + \log\log T\right)\right) \log n\right)$$

for any sequence of labelings $\boldsymbol{\mu}^1, \ldots, \boldsymbol{\mu}^T \in \{-1, 1\}^n$ *such that* $\mu^t_{i_t} = y^t$ *for all* $t \in [T]$.

Proof. Since Algorithm 1 has a conservative update, we may ignore trials on which no mistake is made and thus, from the point of view of the algorithm, a mistake is made on every trial, we will therefore assume that $T = M$. This will lead to a self-referential mistake bound in terms of the number of mistakes made, which we will then iteratively substitute into itself.

Let $c := \log_2 e$, we will use the fact that $\log_2\left(\frac{1}{1-\frac{x}{y+x}}\right) \leq c\frac{x}{y}$ for $x, y > 0$. We will first optimally tune α to give our tuned mistake bound for a general basis set $\mathcal{E}$, and then derive the bounds for bases $\mathcal{F}_n$ and $\mathcal{B}_n$ respectively. The value of α that minimizes (3.8) is

$$\alpha = \frac{\sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{E}}\left(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}}\right)}{\sum_{t=1}^{T} \frac{1}{\pi^t} + \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{E}}(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}})}, \tag{3.18}$$

which when substituted into the second term of (3.8) gives

$$M_{\mathcal{E}} \leq \frac{1}{\pi^1} \log|\mathcal{E}| + c \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{E}}\left(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}}\right) + \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{E}}\left(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}}\right) \log\frac{|\mathcal{E}|}{\alpha}. \tag{3.19}$$

We now upper bound $\frac{1}{\alpha}$ for substitution in the last term of (3.19) for bases $\mathcal{F}_n$ and $\mathcal{B}_n$ separately.

Basis $\mathcal{F}_n$: For $\mathcal{F}_n$ observe that $|\mathcal{E}| = n^2+n$, and since any labeling $\boldsymbol{\mu} \in \{-1, 1\}^n$ of cut-size $\Phi_{\mathcal{S}}(\boldsymbol{\mu})$ is covered by $\Phi_{\mathcal{S}}(\boldsymbol{\mu})+1$ specialists, we have that $\pi^t = \frac{1}{\Phi_{\mathcal{S}}(\boldsymbol{\mu}^t)+1}$ on all trials. We let the number of mistakes made by SCS with basis $\mathcal{F}_n$ be

denoted by $M_{\mathcal{F}_n}$. Thus (3.19) immediately becomes

$$M_{\mathcal{F}_n} \leq \left(\Phi^1 + 1\right) \log |\mathcal{F}_n| + c \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{F}_n}\left(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}}\right) + \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{F}_n}\left(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}}\right) \log\left(\frac{|\mathcal{F}_n|}{\alpha}\right). \tag{3.20}$$

To upper bound $\frac{1}{\alpha}$ we note that if $\boldsymbol{u}^{k_i} \neq \boldsymbol{u}^{k_{i+1}}$ then $J_{\mathcal{F}_n}\left(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}}\right) \geq 1$, and that for $\mathcal{F}_n$, $\frac{1}{\pi_i} = \Phi^{k_i} + 1 \leq n$, thus from (3.18) we have

$$\begin{aligned}
\frac{1}{\alpha} &= 1 + \frac{\sum_{t=1}^{T} \frac{1}{\pi^t}}{\sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{F}_n}(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}})} \\
&\leq 1 + \frac{nT}{|\mathcal{K}| - 1} \\
&= \frac{nT + |\mathcal{K}| - 1}{|\mathcal{K}| - 1} \\
&\leq \frac{(n+1)T}{|\mathcal{K}| - 1}.
\end{aligned}$$

Substituting $\frac{1}{\alpha} \leq \frac{(n+1)T}{|\mathcal{K}|-1}$ into (3.20) gives

$$M_{\mathcal{F}_n} \leq \left(\Phi^1 + 1\right) \log |\mathcal{F}_n| + \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{F}_n}\left(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}}\right) \left[\log\left(e|\mathcal{F}_n|\right) + \log\left(n+1\right) + \log\left(\frac{T}{|\mathcal{K}| - 1}\right)\right]. \tag{3.21}$$

We now exploit the fact that our algorithm is conservative which will allow us to reduce the $\log T$ term in our bound to $\log \log T$ by substituting the self-referential mistake bound into itself. We first simplify (3.21) and substitute

$T = M_{\mathcal{F}_n}$ to give

$$\begin{aligned} M_{\mathcal{F}_n} &\leq \underbrace{\left(\Phi^1 + 1\right) \log |\mathcal{F}_n| + \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{F}_n}\left(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}}\right) \log \left(\frac{e|\mathcal{F}_n|(n+1)}{|\mathcal{K}| - 1} \right)}_{=:\mathcal{Z}} \\ &\quad + \underbrace{\sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{F}_n}\left(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}}\right)}_{=:\mathcal{J}} \log M_{\mathcal{F}_n} \\ &\leq \mathcal{Z} + \mathcal{J} \log \left(\mathcal{Z} + \mathcal{J} \log M_{\mathcal{F}_n} \right) \\ &\leq \mathcal{Z} + \mathcal{J} \log \mathcal{Z} + \mathcal{J} \log \mathcal{J} + \mathcal{J} \log \log M_{\mathcal{F}_n} , \end{aligned}$$

using $\log (a + b) \leq \log (a) + \log (b)$ for $a, b \geq 2$. We finally use the fact that $\mathcal{J} = \mathcal{O}(n|\mathcal{K}|)$ to give $\mathcal{J} \log \mathcal{J} = \mathcal{O}(\mathcal{J} \log (n|\mathcal{K}|))$ and similarly

$$\begin{aligned} \mathcal{J} \log \mathcal{Z} &= \mathcal{O}(\mathcal{J} \log (\Phi^1 \log n + \mathcal{J} \log n)) \\ &= \mathcal{O}(\mathcal{J} \log ((n + \mathcal{J}) \log n))) \\ &= \mathcal{O}(\mathcal{J} \log (n + \mathcal{J})) \\ &= \mathcal{O}(\mathcal{J} \log (n|\mathcal{K}|)) , \end{aligned}$$

to give

$$M_{\mathcal{F}_n} \leq \mathcal{O}\left(\Phi^1 \log n + \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{F}_n}\left(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}}\right) \left(\log n + \log |\mathcal{K}| + \log \log T\right) \right) .$$

Basis $\mathcal{B}_n$: For $\mathcal{B}_n$ we apply the same technique as above, but first observe the following. Without loss of generality assume $n = 2^r$ for some integer r, we then have $|\mathcal{E}| = 4n - 2$. We let the number of mistakes made by SCS with basis $\mathcal{B}_n$

be denoted by $M_{\mathcal{B}_n}$. Thus for basis $\mathcal{B}_n$ (3.19) becomes

$$M_{\mathcal{B}_n} \leq 2\log\frac{n}{2}\left(\Phi^1+1\right)\log|\mathcal{B}_n| + c\sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{B}_n}\big(\boldsymbol{u}^{k_i},\boldsymbol{u}^{k_{i+1}}\big) + \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{B}_n}\big(\boldsymbol{u}^{k_i},\boldsymbol{u}^{k_{i+1}}\big)\log\frac{|\mathcal{B}_n|}{\alpha}\,. \tag{3.22}$$

Recall proposition 7 (that $|\mathcal{C}_{\boldsymbol{\mu}}| \leq 2\log_2\left(\frac{n}{2}\right)(\Phi_{\mathcal{S}}(\boldsymbol{\mu})+1)$) and since $\pi^t = \frac{1}{|\mathcal{C}_{\boldsymbol{\mu}}|}$, then for any labeling $\boldsymbol{\mu}^t \in \{-1,1\}^n$ of cut-size $\Phi_{\mathcal{S}}(\boldsymbol{\mu}^t)$ we have

$$\frac{1}{2(\Phi_{\mathcal{S}}(\boldsymbol{\mu}^t)+1)\log\frac{n}{2}} \leq \pi^t \leq \frac{1}{\Phi_{\mathcal{S}}(\boldsymbol{\mu}^t)+1}\,.$$

We then apply the same argument upper bounding $\frac{1}{\alpha}$,

$$\begin{aligned}\frac{1}{\alpha} &= 1 + \frac{\sum_{t=1}^{T}\frac{1}{\pi^t}}{\sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{B}_n}(\boldsymbol{u}^{k_i},\boldsymbol{u}^{k_{i+1}})}\\ &\leq 1 + \frac{2n\log\left(\frac{n}{2}\right)T}{|\mathcal{K}|-1}\\ &\leq \frac{2n\log\left(\frac{n}{2}\right)T + |\mathcal{K}|-1}{|\mathcal{K}|-1}\\ &\leq \frac{\left(2n\log\left(\frac{n}{2}\right)+1\right)T}{|\mathcal{K}|-1}\,,\end{aligned}$$

and substituting $\frac{1}{\alpha} \leq \frac{(2n\log(\frac{n}{2})+1)T}{|\mathcal{K}|-1}$ into the last term of (3.22) gives

$$M_{\mathcal{B}_n} \leq 2\log_2\left(\frac{n}{2}\right)\left(\Phi^1+1\right)\log|\mathcal{B}_n| + \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{B}_n}\big(\boldsymbol{u}^{k_i},\boldsymbol{u}^{k_{i+1}}\big)\left[c+\log|\mathcal{B}_n|+\ln 2n+\log\left(\frac{T}{|\mathcal{K}|-1}\right)+\log\log n\right].$$

Applying the same recursive technique as above yields a bound of

$$M_{\mathcal{B}_n} \leq \mathcal{O}\left(\Phi^1(\log n)^2 + \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{B}_n}\big(\boldsymbol{u}^{k_i},\boldsymbol{u}^{k_{i+1}}\big)\left(\log n+\log|\mathcal{K}|+\log\log T\right)\right).$$

Using the same argument given in proposition 7 for any two labelings $\boldsymbol{\mu}, \boldsymbol{\mu}' \in \{-1, 1\}^n$, for two consistent well-formed comparators $\boldsymbol{u}, \boldsymbol{u}' \in \Delta_{|\mathcal{B}_n|}$ respectively, and for two consistent well-formed comparators $\hat{\boldsymbol{u}}, \hat{\boldsymbol{u}}' \in \Delta_{|\mathcal{F}_n|}$, we have that $J_{\mathcal{B}_n}(\boldsymbol{u}, \boldsymbol{u}') \leq 2 \log \left(\frac{n}{2}\right) J_{\mathcal{F}_n}(\hat{\boldsymbol{u}}, \hat{\boldsymbol{u}}')$. Finally we use $J_{\mathcal{F}_n} \leq 2H(\boldsymbol{\mu}, \boldsymbol{\mu}')$ from Proposition 11 to complete the proof. □

Thus the bounds are equivalent up to a factor of $\log n$, although the computation times vary dramatically.

We now briefly consider arguments towards a lower bound for this problem. Note that tight upper and lower bounds were proven for graph label prediction on trees in [26]. We give a sketch of a simple argument for a lower bound on the number of mistakes made for predicting a switching sequence of labelings on $\mathcal{S}$. We first describe how introducing and removing cuts can force mistakes in the simplest case.

Given a single graph-labeling problem on $\mathcal{S}$ with a cut-size $\Phi(\boldsymbol{\mu}) = 1$, it is not difficult for an adversary to force $\mathcal{O}(\log n)$ mistakes. Thus in the switching case if $\mathcal{S}$ is uniformly labeled, and a single cut is introduced, then the `learner` can be forced to make $\mathcal{O}(\log n)$ mistakes. Additionally, when switching from a labeling on $\mathcal{S}$ with cut-size $\Phi(\boldsymbol{\mu}) = 2$, to a uniform labeling, a single mistake can be forced.

Now for a switching sequence of graph labelings, $\boldsymbol{\mu}^1, \ldots, \boldsymbol{\mu}^T$, let $\Phi(\boldsymbol{\mu}^t) \ll n$ for all t. For a labeling $\boldsymbol{\mu}$, $\mathcal{S}$ can be divided into $\Phi(\boldsymbol{\mu}) + 1$ segments of length $\frac{n}{\Phi(\boldsymbol{\mu})+1}$. Each segment can be made independent of one another by fixing the boundary vertices between segments. We therefore have $\Phi(\boldsymbol{\mu}) + 1$ independent learning problems, and an adversary can force $\Theta(\log\left(\frac{n}{\Phi(\boldsymbol{\mu})}\right))$ mistakes for every cut introduced and 1 mistake for every 2 cuts removed.

While the bounds in Corollary 12 reflect the *smoothness* of the sequence of labelings, we pay $\mathcal{O}(\log n + \log |\mathcal{K}| + \log \log T)$ for every cut removed *and* introduced for basis set $\mathcal{F}_n$, with an additional logarithmic factor for basis $\mathcal{B}_n$. There is therefore an interesting gap between these bounds and the sketched lower bound, not least of which caused by the $\log \log T$ term, which

we conjecture should be possible to remove.

Note that we may avoid the issue of needing to optimally tune α using the following method proposed by [47] and by [33]. We use a time-varying parameter and on trial t we set $\alpha_t = \frac{1}{t+1}$. We have the following guarantee for this method.

Proposition 13. *For a connected n-vertex graph $\mathcal{G}$ and with randomly sampled spine $\mathcal{S}$, the SCS algorithm with bases $\mathcal{F}_n$ and $\mathcal{B}_n$ in predicting the online sequence $(i_1, y^1), \ldots, (i_T, y^T)$ now with time-varying α set equal to $\frac{1}{t+1}$ on trial t achieves the same asymptotic mistake bounds as in Corollary 12 with an optimally-tuned α, under the assumption that $\Phi_{\mathcal{S}}(\boldsymbol{\mu}^1) \leq \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{E}}(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}})$.*

Proof. Using a time-dependent α we can re-write (3.8) as

$$M_{\mathcal{E}} \leq \frac{1}{\pi^1} \log |\mathcal{E}| + \sum_{t=1}^{T} \frac{1}{\pi^t} \log \left(\frac{1}{1-\alpha_t} \right) + \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{E}}\big(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}}\big) \log \left(\frac{|\mathcal{E}|}{\alpha_{k_{i+1}}} \right), \quad (3.23)$$

and letting $\alpha_t := \frac{1}{t+1}$, and letting $c := \log_2 e$, gives the following,

$$\begin{aligned} M_{\mathcal{E}} &\leq \frac{1}{\pi^1} \log |\mathcal{E}| + \sum_{t=1}^{T} \frac{1}{\pi^t} \log \left(\frac{1}{1-\frac{1}{t+1}} \right) \\ &\quad + \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{E}}\big(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}}\big) \log \left(|\mathcal{E}| \left(k_{i+1} + 1 \right) \right) && (3.24) \\ &\leq \frac{1}{\pi^1} \log |\mathcal{E}| + c \sum_{t=1}^{T} \frac{1}{\pi^t} \frac{1}{t} + \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{E}}\big(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}}\big) \log \left(|\mathcal{E}| T \right) && (3.25) \\ &\leq \frac{1}{\pi^1} \log |\mathcal{E}| + c \left(\max_{t \in [T]} \frac{1}{\pi^t} \right) \sum_{t=1}^{T} \frac{1}{t} + \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{E}}\big(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}}\big) \log \left(|\mathcal{E}| T \right) && (3.26) \\ &\leq \frac{1}{\pi^1} \log |\mathcal{E}| + \left(\max_{t \in [T]} \frac{1}{\pi^t} \right) \log (eT) + \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{E}}\big(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}}\big) \log \left(|\mathcal{E}| T \right) && (3.27) \end{aligned}$$

where the step from (3.24) to (3.25) has used $\log_2 (1+x) \leq cx$ for $x > 0$, and the step from (3.26) to (3.27) has used $\sum_{t \in [T]} \frac{1}{t} < \int_1^T \frac{1}{t} dt + 1 = \ln (eT) = \frac{1}{c} \log_2 (eT)$.

We now use the following upper bound on $\max_{t\in[T]} \frac{1}{\pi^t}$,

$$\max_{t\in[T]} \frac{1}{\pi^t} \leq \frac{1}{\pi^1} + \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{E}}(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}}),$$

and the assumption that $\sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{E}}(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}}) \geq \frac{1}{\pi^1}$, to give

$$\max_{t\in[T]} \frac{1}{\pi^t} \leq 2 \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{E}}(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}}). \tag{3.28}$$

Substituting (3.28) into (3.27) then gives

$$\begin{aligned} M_{\mathcal{E}} &\leq \frac{1}{\pi^1} \log|\mathcal{E}| + 2 \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{E}}\big(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}}\big) \left(\log(eT) + \frac{1}{2} \log(|\mathcal{E}|T) \right) \\ &= \frac{1}{\pi^1} \log|\mathcal{E}| + 2 \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{E}}\big(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}}\big) \left(\frac{1}{2} \log(|\mathcal{E}|) + \log(e) + \frac{3}{2} \log(T) \right). \end{aligned}$$

Using a conservative update (see section 3.5), we similarly set $\alpha_t := \frac{1}{m+1}$, where m is the current number of mistakes of the algorithm. We next use the same "recursive trick" as that in the proof of Corollary 12. The proof follows analogously, leaving

$$M_{\mathcal{F}_n} \leq \mathcal{O}\left(\Phi^1 \log n + \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{F}_n}\big(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}}\big) (\log n + \log|\mathcal{K}| + \log\log T) \right)$$

for the basis set $\mathcal{F}_n$, and

$$M_{\mathcal{B}_n} \leq \mathcal{O}\left(\Phi^1 (\log n)^2 + \sum_{i=1}^{|\mathcal{K}|-1} J_{\mathcal{B}_n}\big(\boldsymbol{u}^{k_i}, \boldsymbol{u}^{k_{i+1}}\big) (\log n + \log|\mathcal{K}| + \log\log T) \right)$$

for the basis set $\mathcal{B}_n$. □

Algorithm 2 SWITCHING GRAPH PERCEPTRON

Input: Graph $\mathcal{G}$; $\gamma > 0$
Initialize: $\boldsymbol{w}^1 \leftarrow \mathbf{0}$; $\boldsymbol{K} \leftarrow \boldsymbol{L}_{\mathcal{G}}^{+} + \max_{i \in [n]} (\boldsymbol{e}_i^\top \boldsymbol{L}_{\mathcal{G}}^{+} \boldsymbol{e}_i) \mathbf{1}\mathbf{1}^\top$

for $t \leftarrow 1$ to T **do**
  receive $i_t \in V$
  predict $\hat{y}^t \leftarrow \text{sign}(w_{i_t}^t)$
  receive $y^t \in \{-1, 1\}$
  if $\hat{y}^t \neq y^t$ **then**
    $\dot{\boldsymbol{w}}^t \leftarrow \boldsymbol{w}^t + y^t \frac{\boldsymbol{K}\boldsymbol{e}_{i_t}}{\boldsymbol{K}_{i_t,i_t}}$
    if $\|\dot{\boldsymbol{w}}^t\|_{\boldsymbol{K}} > \gamma$ **then**
      $\boldsymbol{w}^{t+1} \leftarrow \frac{\dot{\boldsymbol{w}}^t}{\|\dot{\boldsymbol{w}}^t\|_{\boldsymbol{K}}} \gamma$ ▷ Projection step
    else
      $\boldsymbol{w}^{t+1} \leftarrow \dot{\boldsymbol{w}}^t$
    end if
  else
    $\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t$
  end if
end for

3.6 The Switching-Graph Perceptron

Before we present results of experiments of predicting switching graph labelings, we first introduce an additional algorithm that addresses the graph-switching problem using a variant of the Kernel Perceptron which will serve as a benchmark in our experiments in Section 3.7. This algorithm is not novel to this work. Indeed, this approach and corresponding mistake bound was first sketched out for the switching graph labeling setting in [27, Sec. 6.2]. For the sake of completeness we include the algorithm here (Algorithm 2) and give a proof of its mistake bound in Appendix A.

In Chapter 4 we will study adaptive algorithms that employ *projection* updates, primarily building on the work of [37]. In that paper the switching perceptron algorithm is presented, which forms the basis of Algorithm 2. Intuitively the approach controls the weights of the algorithm by projecting the weight vector onto a suitable convex set, Γ. In this case, a sphere of radius γ centered at the origin.

The key to the approach in the switching graph labeling setting is to use

Algorithm	Time	Mistake Bound
SCS-F	$\mathcal{O}(n^2)$	$\mathcal{O}\left(\sum_{k\in\mathcal{K}} \Phi^k \left(\log n + \log\lvert\mathcal{K}\rvert + \log\log T\right)\right)$
SCS-B	$\mathcal{O}(\log n)$	$\mathcal{O}\left(\left(\sum_{k\in\mathcal{K}} \Phi^k \left(\log n + \log\lvert\mathcal{K}\rvert + \log\log T\right)\right)\log n\right)$
SGP	$\mathcal{O}(n)$	$\mathcal{O}\left(\sum_{k\in\mathcal{K}} \sqrt{\Phi^k\hat{\Phi}}R_{\mathcal{G}}\right)$

Table 3.1: Mistake bounds and per-trial time complexities of the algorithms given for predicting switching graph labelings.

the following graph kernel (introduced by [29])

$$\boldsymbol{K} := \boldsymbol{L}_{\mathcal{G}}^{+} + R_L \mathbf{1}\mathbf{1}^\top$$

where $\boldsymbol{L}_{\mathcal{G}}^{+}$ denotes the pseudo-inverse of the graph Laplacian, and $R_L := \max_i(\boldsymbol{e}_i^\top \boldsymbol{L}_{\mathcal{G}}^{+}\boldsymbol{e}_i)$,. The norm induced by this kernel is denoted $\|\boldsymbol{\mu}\|_{\boldsymbol{K}} := \sqrt{\boldsymbol{\mu}^\top \boldsymbol{K}^{-1}\boldsymbol{\mu}}$, for which we have the following properties: $\max_i(\boldsymbol{e}_i^\top \boldsymbol{K}\boldsymbol{e}_i) \leq 2R_{\mathcal{G}}$ and if $\boldsymbol{\mu}\in\{-1,1\}^n$ then $\|\boldsymbol{\mu}\|_{\boldsymbol{K}}^2 = \Theta(\Phi(\boldsymbol{u}))$ (for details see [29]). The following theorem then follows from refinements of these properties and [37, Theorem 10].

Theorem 14. *The* Switching Graph Perceptron *(see Algorithm 2) with kernel* $\boldsymbol{K} = \boldsymbol{L}_{\mathcal{G}}^{+} + R_L\mathbf{1}\mathbf{1}^\top$ *and* $R_L := \max_i(\boldsymbol{e}_i^\top \boldsymbol{L}_{\mathcal{G}}^{+}\boldsymbol{e}_i)$*, and radius parameter* $\gamma = \max_t \|\boldsymbol{\mu}^t\|_{\boldsymbol{K}}$ *incurs no more than*

$$\mathcal{O}\left(\sum_{k\in K} \sqrt{\Phi^k\hat{\Phi}}R_{\mathcal{G}}\right)$$

mistakes in predicting the online sequence $(i_1, y^1), \ldots, (i_T, y^T)$ *for any sequence of labelings* $\boldsymbol{\mu}^1, \ldots, \boldsymbol{\mu}^T \in \{-1, 1\}^n$ *such that* $\mu_{i_t} = y^t$ *for all* $t \in [T]$.

For completeness we provide a proof of Theorem 14 in Appendix A. Note that a naive computation of $\|\dot{\boldsymbol{w}}^t\|_{\boldsymbol{K}}$ would require $\mathcal{O}(n^2)$ time, however, since

after the Perceptron makes a mistake we have

$$\|\dot{\boldsymbol{w}}^t\|_{\boldsymbol{K}}^2 = \|\boldsymbol{w}^t + y^t \frac{\boldsymbol{v}_{i_t}}{\|\boldsymbol{v}_{i_t}\|_{\boldsymbol{K}}^2}\|_{\boldsymbol{K}}^2 = \|\boldsymbol{w}^t\|_{\boldsymbol{K}}^2 + \frac{1}{\|\boldsymbol{v}_{i_t}\|_{\boldsymbol{K}}^2} + \frac{2y^t w_{i_t}}{\|\boldsymbol{v}_{i_t}\|_{\boldsymbol{K}}^2}\,, \tag{3.29}$$

where $\boldsymbol{v}_{i_t} := \boldsymbol{K}\boldsymbol{e}_{i_t}$, and $\|\boldsymbol{v}_{i_t}\|_{\boldsymbol{K}}^2 = \boldsymbol{e}_{i_t}^\top \boldsymbol{K}\boldsymbol{K}^{-1}\boldsymbol{K}\boldsymbol{e}_{i_t} = \boldsymbol{K}_{i_t,i_t}$. The value of $\|\dot{\boldsymbol{w}}^t\|_{\boldsymbol{K}}$ can therefore be updated on each trial in $\mathcal{O}(1)$ time. The SWITCHING GRAPH PERCEPTRON algorithm therefore has $\mathcal{O}(n)$ time complexity.

3.7 Experimental Results

In this section we present results of experiments on real data. At the time of data collection, the city of Chicago contained 608 public bicycle stations for its "Divvy Bike" sharing system. Current and historical data is available from the City of Chicago[3] containing a variety of features for each station, including latitude, longitude, number of docks, number of operational docks, and number of docks occupied. The latest data on each station is published approximately every ten minutes.

We used a sample of 72 hours of data, from $4:55am$ on 8^{th} April 2019 to 4 : 55am on 11^{th} April 2019. The first 24 hours of data were used for parameter selection, and the remaining 48 hours of data were used for evaluating performance. Any stations that were not in service during any of the 72 hours were removed from the dataset (in this case there was only one such station). On each ten-minute snapshot we took the percentage of empty docks of each station. We created a binary labeling from this data by setting a threshold of 50%. Thus each bicycle station is a vertex in our graph, and the label of each vertex indicates whether that station is "mostly full" or "mostly empty." See Figure 3.7 for two example labelings. Due to this thresholding the labels of some "quieter" stations were observed not to switch, as the percentage of available docks rarely changed. These stations tended to be on the "outskirts," and thus we excluded these stations from our experiments, giving 404 vertices in our graph.

[3] https://data.cityofchicago.org/Transportation/Divvy-Bicycle-Stations-Historical/eq45-8inv

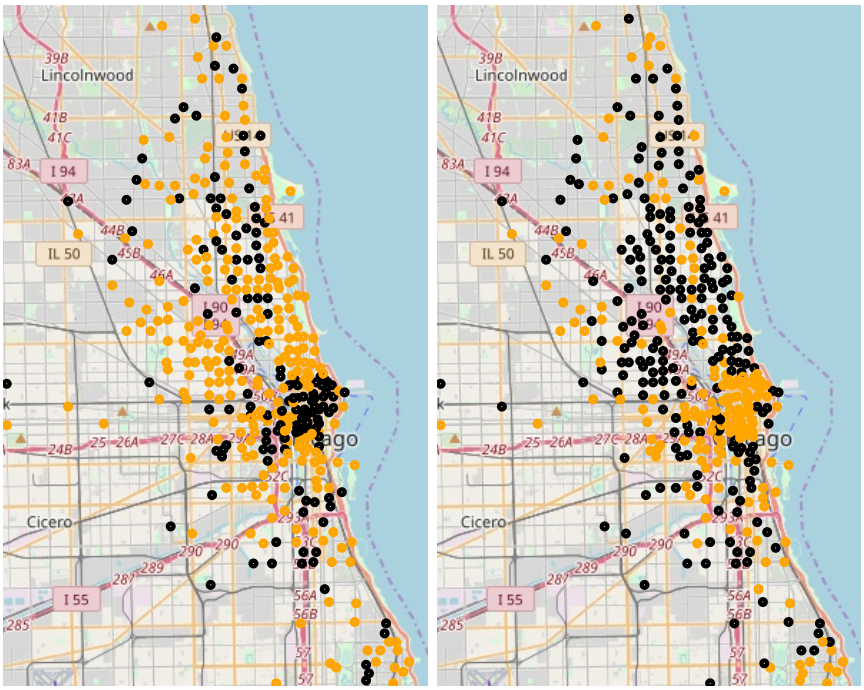


Figure 3.7: An example of two binary labelings taken from the morning and evening of the first 24 hours of data. An 'orange' label implies that station is $< 50\%$ full and a 'black' label implies that station is $\geq 50\%$ full.

Using the geodesic distance between each station's latitude and longitudinal position a connected graph was built using the union of a k-nearest neighbor graph ($k = 3$) and a minimum spanning tree. For each instance of our algorithm the graph was then transformed in the manner described in Section 3.3, by first drawing a spanning tree uniformly at random and then linearizing using depth-first search.

As natural benchmarks for this setting we considered the following four methods of prediction:

1. "**Global**": For all vertices predict with the most frequently occurring label of the entire graph from the training data.

2. "**Local**": For each vertex predict with its most frequently occurring label from the training data.

3. "**Temporal-Global**": For all vertices at any given time predict with the most frequently occurring label of the entire graph at that time from the

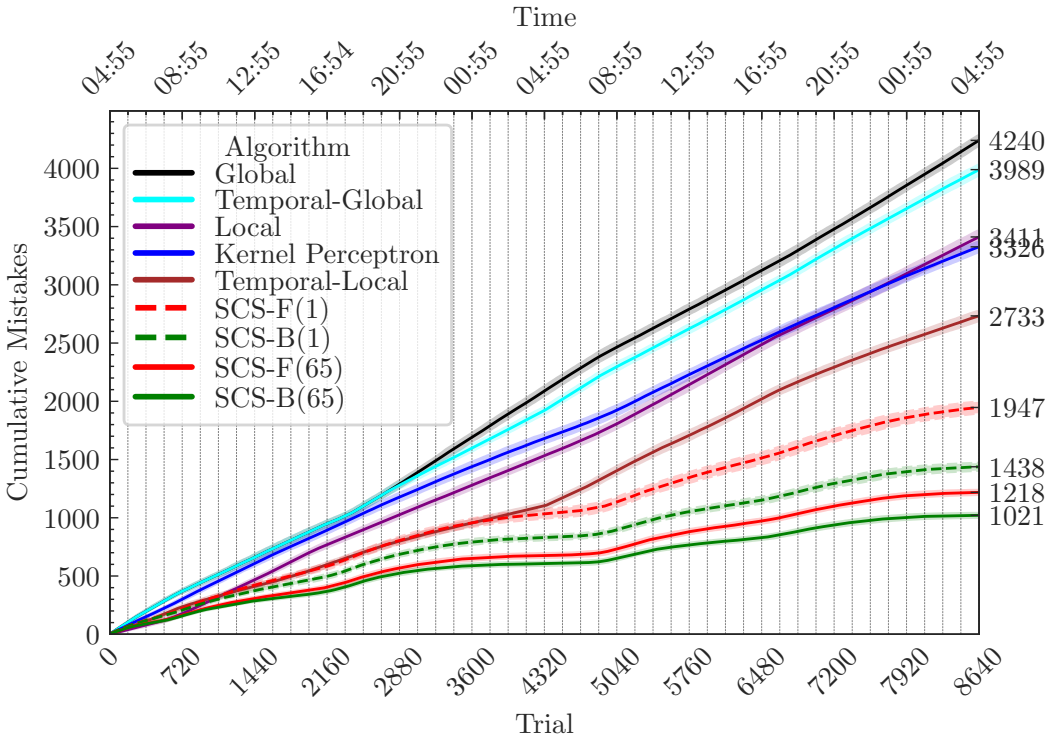


Figure 3.8: Mean cumulative mistakes over 25 iterations for all algorithms and benchmarks over 48 hours (8640 trials) on a 404-vertex graph. A comparison of the mean performance of SCS with bases $\mathcal{F}_n$ and $\mathcal{B}_n$ (SCS-F and SCS-B respectively) using an ensemble of size 1 and 65 is shown. Error bars represent one standard deviation.

training data.

4. "**Temporal-Local**": For each vertex at any given time predict with that vertex's label observed at the same time in the training data.

We also compare our algorithms against the kernel Perceptron described in Section 3.6.

Following the experiments of [18] in which ensembles of random spanning trees were drawn and aggregated by an unweighted majority vote, we tested the effect on performance of using ensembles of instances of our algorithms, aggregated in the same fashion. We tested ensemble sizes in $\{1, 3, 5, 9, 17, 33, 65\}$, using odd numbers to avoid ties.

For every ten-minute snapshot (labeling) we queried 30 vertices uniformly at random (with replacement) in an online fashion, giving a sequence of 8640

Table 3.2: Mean error ± std over 25 iterations on a 404-vertex graph for all algorithms and benchmarks, and for all ensemble sizes of SCS-Fand SCS-B.

Algorithm	Ensemble Size 1	3	5	9	17	33	65
SCS-F	1947 ± 49	1597 ± 32	1475 ± 30	1364 ± 28	1293 ± 26	1247 ± 21	1218 ± 19
SCS-B	1438 ± 32	1198 ± 27	1127 ± 25	1079 ± 24	1050 ± 23	1032 ± 22	1021 ± 18
Kernel Perceptron	3326 ± 43	-	-	-	-	-	-
Local	3411 ± 55	-	-	-	-	-	-
Global	4240 ± 44	-	-	-	-	-	-
Temporal (Local)	2733 ± 42	-	-	-	-	-	-
Temporal (Global)	3989 ± 44	-	-	-	-	-	-

Table 3.3: Parameter ranges used for optimizing the three algorithms with tunable parameters.

Algorithm	Parameter	Range	Optimal Value
Kernel Perceptron	γ	$3.5 - 5$	3.89
SCS-F	α	$1 \times 10^{-12} - 1 \times 10^{-6}$	7.4×10^{-10}
SCS-B	α	$1 \times 10^{-5} - 5 \times 10^{-4}$	3.0×10^{-4}

trials over 48 hours. The average performance over 25 iterations is shown in Figure 3.8. There are several surprising observations to be made from our results. Firstly, both SCS algorithms performed significantly better than all benchmarks and competing algorithms. Additionally basis $\mathcal{B}_n$ outperformed basis $\mathcal{F}_n$ by quite a large margin, despite having the weaker bound and being exponentially faster. Finally, we observed a significant increase in performance of both SCS algorithms by increasing the ensemble size (see Figure 3.8).

Table 3.3 shows the parameter ranges searched over for the two variants of our SCS algorithm, and the kernelized Perceptron algorithm. Parameters were tuned using an exhaustive search over the ranges specified in Table 3.3, taking the mean minimizer over 10 iterations.

Interestingly when tuning α we found basis $\mathcal{B}_n$ to be very robust, while $\mathcal{F}_n$ was very sensitive. This observation combined with the logarithmic per-trial time complexity suggests that SCS with $\mathcal{B}_n$ has promise to be a very practical algorithm.

Chapter 4

Improved Regret Bounds for Tracking Experts with Memory

4.1 Introduction to the Chapter

In this chapter we address the problem of sequential prediction with expert advice [7] in a non-stationary environment with long-term memory guarantees in the sense of Bousquet and Warmuth [1]. We introduced the prediction with expert advice setting in Chapter 2. Recall that in this model `nature` sequentially generates outcomes which the `learner` attempts to predict. Before making each prediction, the `learner` listens to a set of n experts who each make their own predictions. The `learner` bases its prediction on the advice of the experts. After the prediction is made and the true outcome is revealed by `nature`, the accuracies of the `learner`'s prediction and the expert predictions are measured by a loss function. The `learner` receives information on all expert losses on each trial. We make no distributional assumptions about the outcomes generated, indeed `nature` may be assumed to be adversarial. The goal of the `learner` is to predict well relative to a predetermined comparison class of predictors, in this case the set of experts themselves. Unlike the standard regret model, where the `learner`'s performance is compared to the single best predictor in hindsight, our aim is for the `learner` to predict well relative to a sequence of comparison predictors. That is, "switches" occur in the data

sequence and different experts are assumed to predict well at different times.

In this work our focus is on the case when this sequence consists of a few unique predictors relative to the number of switches. Thus most switches return to a previously "good" expert, and a `learner` that can exploit this fact by "remembering" the past can adapt more quickly than a `learner` who has no memory and must re-learn the experts after every switch. The problem of switching with memory in online learning is part of a much broader and fundamental problem in machine learning: how a system can adapt to new information yet retain knowledge of the past. This is an area of research in many fields, including for example, catastrophic forgetting in artificial neural networks [48, 49].

The chapter is organized as follows. We first introduce the model and discuss related work, giving a detailed overview of the previous results on which we improve. In Section 4.3 we give our main results, a regret bound which holds for two algorithms, and an algorithm to compute relative entropy projection with non-uniform lower box constraints in linear time. In Section 4.5 we derive a new "geometric-decay mixing scheme" for the first experts memory algorithm (Mixing Past Posteriors [1]), and show the correspondence to the current best known algorithm [2].

4.1.1 Notation

We first define the following additional notation which is specific to this chapter. Let $\Delta_n^\alpha := \{\boldsymbol{u} \in [0, \alpha]^n : \|\boldsymbol{u}\|_1 = \alpha\}$ be a scaled simplex. We define ri S to be the relative interior of the set S. For two vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ we say $\boldsymbol{\alpha} \preceq \boldsymbol{\beta}$ if and only if $\alpha_i \leq \beta_i$ for all $i = 1, \ldots, n$, and $\boldsymbol{\alpha} \succeq \boldsymbol{\beta}$ if and only if $\alpha_i \geq \beta_i$ for all $i = 1, \ldots, n$.

4.2 Background and Related Work

We now formally introduce the model of sequential prediction with expert advice. In this setting `nature` generates elements from an outcome space, $\mathcal{Y}$ while the predictions of the `learner` and the experts are elements from a

prediction space, $\mathcal{D}$. Given a non-negative loss function $\ell : \mathcal{D} \times \mathcal{Y} \to [0, \infty]$, learning proceeds in trials. On each trial $t = 1, \ldots, T$: 1) the `learner` receives the expert predictions $\boldsymbol{x}^t \in \mathcal{D}^n$, 2) the `learner` makes a prediction $\hat{y}^t \in \mathcal{D}$, 3) `nature` reveals the true label $y^t \in \mathcal{Y}$, and 4) the `learner` suffers loss $\ell^t := \ell(\hat{y}^t, y^t)$ and expert i suffers loss $\ell_i^t := \ell(x_i^t, y^t)$ for $i = 1, \ldots, n$. Common to all of the algorithms we will consider in this chapter is a weight vector, $\boldsymbol{w}^t \in \Delta_n$, where w_i^t can be interpreted as the algorithm's confidence in expert i on trial t. In this work we assume that the `learner` uses a prediction function $\texttt{pred} : \Delta_n \times \mathcal{D}^n \to \mathcal{D}$ to generate its prediction $\hat{y}^t = \texttt{pred}(\boldsymbol{w}^t, \boldsymbol{x}^t)$ on trial t. A classic example is to predict with the weighted average of the expert predictions, that is, $\texttt{pred}(\boldsymbol{w}^t, \boldsymbol{x}^t) = \boldsymbol{w}^t \cdot \boldsymbol{x}^t$, although for some loss functions improved bounds are obtained with different prediction functions (see e.g., [50]). In this chapter we will assume (c, η)-realizability of ℓ and `pred` [1, 51, 52]. That is, there exists constants $c, \eta > 0$ such that for all $\boldsymbol{w} \in \Delta_n$, $\boldsymbol{x} \in \mathcal{D}^n$, and $y \in \mathcal{Y}$,

$$\ell(\texttt{pred}(\boldsymbol{w}, \boldsymbol{x}), y) \leq -c \ln \left(\sum_{i=1}^{n} w_i e^{-\eta \ell(x_i, y)} \right).$$

This includes η-exp-concave losses when $\texttt{pred}(\boldsymbol{w}^t, \boldsymbol{x}^t) = \boldsymbol{w}^t \cdot \boldsymbol{x}^t$ and $c = \frac{1}{\eta}$. For simplicity we present regret bound guarantees that assume $(c, \frac{1}{c})$-realizability, that is $c\eta = 1$. This includes the log loss with $c = 1$, and the square loss with $c = \frac{1}{2}$ when $\mathcal{D} = \mathcal{Y} = [0, 1]$. For any comparison sequence of experts $i_{1:T} = i_1, \ldots, i_T \in [n]$ the regret of the `learner` with respect to this sequence is defined as

$$\mathcal{R}(i_{1:T}) := \sum_{t=1}^{T} \ell^t - \sum_{t=1}^{T} \ell_{i_t}^t .$$

We consider and derive algorithms which belong to the family of "exponential weights" (EW) algorithms (see e.g., [53, 7, 50]). After receiving the expert losses the EW algorithm applies the following incremental loss update to the expert weights,

$$\dot{w}_i^t = \frac{w_i^t e^{-\eta \ell_i^t}}{\sum_{j=1}^{n} w_j^t e^{-\eta \ell_j^t}} . \tag{4.1}$$

4.2.1 Static setting

In the static setting the `learner` competes against a single expert (i.e., $i_1 = \ldots = i_T$). For the static setting the EW algorithm sets $\boldsymbol{w}^{t+1} = \dot{\boldsymbol{w}}^t$ for the next trial, and for $(c, \frac{1}{c})$-realizable losses and prediction functions achieves a static regret bound of $\mathcal{R}(i_{1:T}) \leq c \ln n$.

4.2.2 Switching

In the switching (without memory) setting the `learner` competes against a sequence of experts $i_1, \ldots, i_T$ with $k := \sum_{t=1}^{T-1} [i_t \neq i_{t+1}]$ switches. The well-known Fixed-Share algorithm [30] solves the switching problem with the update

$$\boldsymbol{w}^{t+1} = (1-\alpha)\dot{\boldsymbol{w}}^t + \alpha \frac{\mathbf{1}}{n}, \tag{4.2}$$

by forcing each expert to "share" a fraction of its weight *uniformly* with all experts.[1] Recall that we used this update in Chapter 3 for achieving switching guarantees with cluster specialists. The update is parameterized by a "switching" parameter, $\alpha \in [0, 1]$. With an optimally-tuned $\alpha = \frac{k}{T-1}$ the regret with respect to the best sequence of experts with k switches is

$$\begin{aligned} \mathcal{R}(i_{1:T}) &\leq c\left((k+1)\ln n + (T-1)\mathcal{H}\left(\frac{k}{T-1}\right)\right) \\ &\leq c\left((k+1)\ln n + k \ln\left(\frac{T-1}{k}\right) + k\right). \end{aligned} \tag{4.3}$$

4.2.3 Switching with Memory

Freund [41] gave an open problem to improve on the regret bound (4.3) when the comparison sequence of experts is comprised of a small pool of size $m := |\cup_{t=1}^{T} \{i_t\}| \ll k$. Using counting arguments Freund gave an exponential-time algorithm with the information-theoretic ideal regret bound

[1] Technically in the original Fixed-Share update each expert shares weight to all *other* experts, i.e., $w_i^{t+1} = (1-\alpha)\dot{w}_i^t + \frac{\alpha}{n-1}\sum_{j \neq i} \dot{w}_j^t$. The two updates achieve essentially the same regret bound and are equivalent up to a scaling of α.

of $\mathcal{R}(i_{1:T}) \leq c \ln \left(\binom{n}{m} \binom{T-1}{k} m(m-1)^k \right)$, which is upper-bounded by

$$c \left(m \ln n + k \ln \left(\frac{T-1}{k} \right) + (k - m + 1) \ln m + k + m \right) . \tag{4.4}$$

The first efficient algorithm solving Freund's problem was presented in the seminal paper [1]. This work introduced the notion of a *mixing scheme*, which is a distribution $\boldsymbol{\gamma}^{t+1}$ with support $\{0, \ldots, t\}$. Given $\boldsymbol{\gamma}^{t+1}$, the algorithm's update on each trial is the *mixture* over all past weight vectors,

$$\boldsymbol{w}^{t+1} = \sum_{q=0}^{t} \gamma_q^{t+1} \dot{\boldsymbol{w}}^q , \tag{4.5}$$

where $\dot{\boldsymbol{w}}^0 := \frac{1}{n}\mathbf{1}$, and $\gamma_0^1 := 1$. Intuitively, by mixing all "past posteriors" (MPP) the weights of previously well-performing experts can be prevented from vanishing and recover quickly. An efficient mixing scheme requiring $\mathcal{O}(n)$-time per trial is the "*uniform*" mixing scheme given by $\gamma_t^{t+1} = 1 - \alpha$ and $\gamma_q^{t+1} = \frac{\alpha}{t}$ for $0 \leq q < t$. A better regret bound was proved with a "*decaying*" mixing scheme, given by

$$\gamma_q^{t+1} = \begin{cases} 1 - \alpha & q = t \\ \alpha \frac{1}{(t-q)^\rho} \frac{1}{Z_t} & 0 \leq q < t , \end{cases} \tag{4.6}$$

where $Z_t = \sum_{q=0}^{t-1} \frac{1}{(t-q)^\rho}$ is a normalizing factor, and $\rho \geq 0$. With a tuning of $\alpha = \frac{k}{T-1}$ and $\rho = 1$ this mixing scheme achieves a regret bound of[2]

$$\mathcal{R}(i_{1:T}) \leq c \left(m \ln n + 2k \ln \left(\frac{T-1}{k} \right) + k \ln (m-1) + k + k \ln \ln (eT) \right) . \tag{4.7}$$

It appeared that to achieve the best regret bounds, the mixing scheme needed to decay towards the past. Unfortunately, computing (4.6) exactly requires the storage of all past weights, at a cost of $\mathcal{O}(nt)$-time and space per trial. Observe that these schemes set $\gamma_t^{t+1} = 1 - \alpha$, where typically α is small, since intuitively switches are assumed to happen infrequently. All updates using such schemes

[2] (4.7) is a simplified upper bound of the bound given in [1, Corollary 9], using $\ln (1 + x) \leq x$.

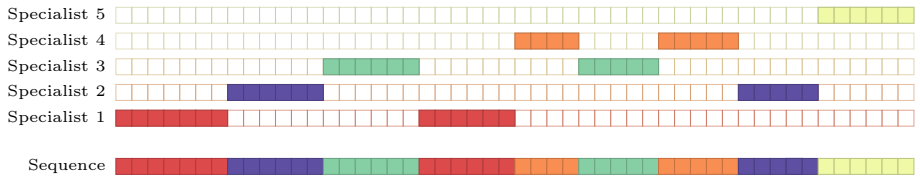


Figure 4.1: An example trial sequence of experts (colors) and the $m = 5$ circadian specialists required to predict exactly as the sequence. A filled square implies a specialist is awake, and predicts according to that expert (color). An empty square impies a specialist is asleep, and abstains from predicting.

are of the form

$$\boldsymbol{w}^{t+1} = (1-\alpha)\dot{\boldsymbol{w}}^t + \alpha \boldsymbol{v}^t\,, \tag{4.8}$$

where $\boldsymbol{v}^t \in \Delta_n$ is a function of the past weights $\dot{\boldsymbol{w}}^0, \ldots, \dot{\boldsymbol{w}}^{t-1}$. We will refer to (4.8) as the *generalized share update* (see [34]). Fixed-Share is a special case when $\boldsymbol{v}^t = \frac{\mathbf{1}}{n}$ for all t. This generalized share update features heavily in this chapter.

For a decade it remained an open problem to give the MPP update a Bayesian interpretation. This was finally solved in [2] with the use of *circadian specialists*.[3] We briefy introduced the notion of specialists in Chapter 2, and used the technique of specialists in Chapter 3 for prediction on graphs. In this context however, given n experts, each specialist on each trial t is either *awake* and predicts in accordance with a prescribed base expert, or is *asleep* and abstains from predicting. For n base experts and finite time horizon T there are therefore $n2^T$ circadian specialists. For Freund's problem an assembly of m circadian specialists can predict exactly as the comparison sequence of experts (see Figure 4.1 for an example sequence).

The Bayesian interpretation of the MPP update given in [2, Theorem 2] was simple: to define a mixing scheme $\boldsymbol{\gamma}^{t+1}$ was to induce a prior over this set of circadian specialists. The authors of [2] proposed a simple Markov chain prior over the set of circadian specialists, giving an efficient $\mathcal{O}(n)$-time per trial

[3] In [2] these specialists are called partition specialists, in this thesis however we refer to them as circadian specialists.

algorithm with the regret bound

$$\mathcal{R}(i_{1:T}) \leq c\left[m\ln\frac{n}{m} + m\mathcal{H}\left(\frac{1}{m}\right) + (T-1)\mathcal{H}\left(\frac{k}{T-1}\right) + (m-1)(T-1)\mathcal{H}\left(\frac{k}{(m-1)(T-1)}\right)\right] \tag{4.9}$$

$$\leq c\left(m\ln n + 2k\ln\frac{T-1}{k} + (k-m+1)\ln m + 2(k+1)\right), \tag{4.10}$$

which is currently the best known regret bound for Freund's problem. In this work we slightly improve on the bound (4.9) for tracking experts with memory (Theorem 15). We also show that in fact this Markov prior on circadian specialists corresponds to a geometrically-decaying mixing scheme for MPP (Proposition 24). The regret bounds discussed in this chapter all rely on optimally tuning one or more parameters, which in practice are usually unknown, and this is true for our regret bound.

Adaptive online learning algorithms with memory have been shown to have better empirical performance than those without memory [54], and to be effective in real-world applications such as intrusion detection systems [55]. While considerable research has been done on switching with memory in online learning (see e.g., [1, 2, 12, 34, 56, 57]), there remain several open problems. Firstly, there remains a gap between the best known regret bound for an efficient algorithm and the information-theoretic ideal bound (4.4). Present in both bounds (4.7) and (4.10) is the factor of 2 in the second term, which does not appear in (4.4). In [2] this was interpreted as the cost of co-ordination between specialists, essentially one "pays" twice per switch as one specialist falls asleep and another awakens. In this chapter we make some small progress towards closing this gap by avoiding such additional costs the first time each expert is learned by the algorithm. That is, we pay to *remember* but not to *learn*.

Secondly, unless n is very large the bound (4.9) beats Fixed-Share's bound (4.3) only when $m \ll k$, but suffers when m is even a moderate fraction of k. A natural question is can we improve on Fixed-Share when we relax the assumption that $m \ll k$, and only a few members of a sequence of experts

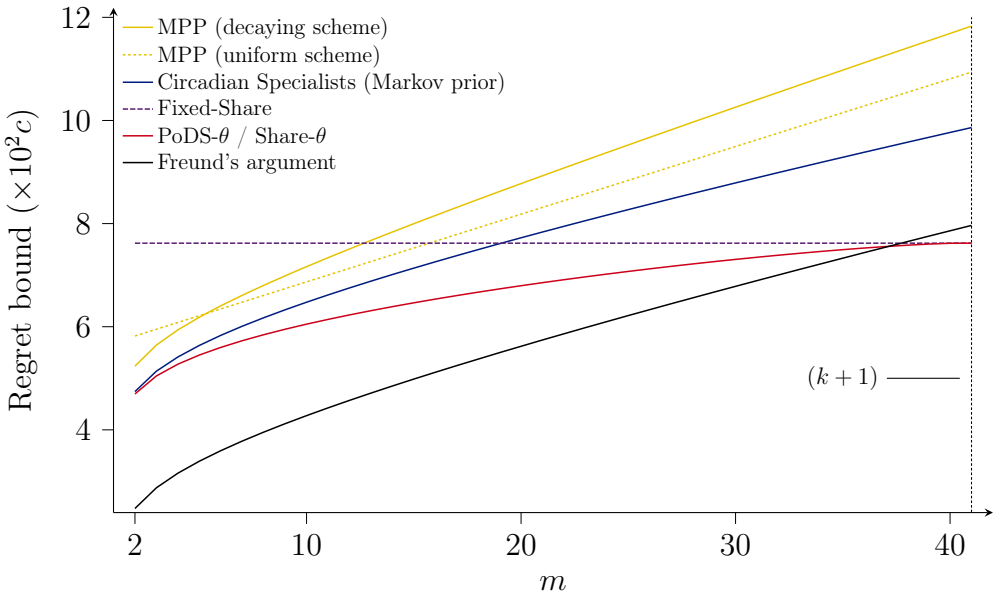


Figure 4.2: A comparison of the regret bounds discussed in this chapter for $m \in [2, k+1]$ with $n = 500000$, $k = 40$, and $T = 4000$. Fixed-Share's bound is constant with respect to m. In this case previous "memory" bounds (blue & yellow) are much worse than Fixed-Share for larger values of m while our bound (red) improves on Fixed-Share for all $m \in [2, k]$ (for sufficiently large n).

need remembering (consider for instance, $m > k/2$)? In this chapter we prove a regret bound that is not only tighter than (4.9) for all m, but for sufficiently large n improves on Fixed-Share for all $m \leq k$. In Figure 4.2 this behavior is shown for several existing regret bounds and our regret bound. Note that as $m \to k$, then the bound (4.4) from Freund's argument is slightly worse than even Fixed Share. This is due to the fact that the quantity $\binom{n}{m}\binom{T-1}{k}m(m-1)^k$ "over counts" the number of sequences with k switches containing m distinct experts.

Our regret bound will hold for two algorithms; one utilizes a weight-sharing update in the sense of (4.8), and the other utilizes a projection update. Why should we consider projections? Consider for example a large model consisting of many weights, and to update these weights costs time and/or money. Alternatively consider the application of regret-bounded adaptive algorithms in online portfolio selection (see e.g., [58, 59]). Here each "expert" corresponds to a single stock and the weight vector $\boldsymbol{w}^t$ corresponds to a

(normalized) portfolio. If ℓ_i^t is the negative log return of stock i after day t, then the loss function $\ell^t := -\ln \sum_{i=1}^n w_i^t e^{-\ell_i^t}$ is the negative log return of the portfolio. This loss is $(1,1)$-realizable by definition (although there is no prediction function [11]), and the daily price changes in the market naturally induce the "loss update" (4.1) by updating the portfolio weights. The algorithm's secondary update (projection or weight-sharing) requires the investor to then actively buy/sell to re-balance the portfolio after each trading period, but doing so may incur transaction costs proportional to the amount bought or sold (see e.g., [60, 58]). In Section 4.4 we motivate the use of projections over weight-sharing in this context, proving that projections are strictly more "efficient." Online portfolio selection with transaction costs is an active area of research [58, 61, 62, 63], and many tools developed for the problem of prediction with expert advice have been used in this area (see e.g., [59, 64, 65, 66]).

4.2.4 Related Work

Switching (without memory) in online learning was first introduced in [7] (see also the earlier [67] and independently in the context of universal coding in [68]), and extended with the Fixed-Share algorithm [30]. An extensive literature has built on these works, including but not limited to [1, 2, 11, 12, 32, 34, 36, 37, 57, 69, 70, 71, 72]. Relevant to this work are the results for switching with memory [1, 2, 12, 34, 57, 70]. The first was the seminal work of [1]. The best known result is given in [2], which we improve on. In [57] a reduction of switching with memory to switching without memory is given, although with a slightly worse regret bound than [1]. Related to the experts model is the *bandits* setting, which was addressed in the memory setting in [57], and we also consider in Chapter 5.

In [34] a unified analysis of both Fixed-Share and MPP was given in the context of online convex optimization. They observed the generalized share update (4.8) and slightly improved the bounds of [1]. Adaptive regret [7, 11, 36, 73] has been used to prove regret bounds for switching but unfortunately does not generalize to the memory setting. This chapter primarily builds on

the work of [1] with a new geometrically-decaying mixing scheme, and on [37] with a new relative entropy projection algorithm. Related to the problem of prediction with expert advice is that of universal coding in information theory (see e.g., [74, 75, 76] for a discussion). Similarly, related to the problem of tracking experts with memory is the problem of universal coding for switching sources with repeating statistics (see e.g., [68, 77, 78] and references therein).

4.3 Projection onto Dynamic Sets (PoDS)

In this section we give a relative entropy projection-based algorithm for tracking experts with memory. We show how the projection update used by this algorithm is intimately related to the generalized share update (4.8). In the following subsection we propose a specific update rule for our algorithm for which we improve on the best known regret bound for this problem.

Given a non-empty set $\mathcal{C} \subseteq \Delta_n$ and a point $\boldsymbol{w} \in \text{ri } \Delta_n$ we define

$$\mathcal{P}(\boldsymbol{w}; \mathcal{C}) := \operatorname*{arg\,min}_{\boldsymbol{u} \in \mathcal{C}} D(\boldsymbol{u}, \boldsymbol{w})$$

to be the projection with respect to the relative entropy of $\boldsymbol{w}$ onto $\mathcal{C}$ [79]. Such projections were first introduced for switching (without memory) in online learning in [37], in which after every trial the weight vector $\dot{\boldsymbol{w}}^t$ is projected onto $\mathcal{C} = [\frac{\alpha}{n}, 1]^n \cap \Delta_n$, that is, the simplex with uniform box constraints. For prediction with expert advice this projection algorithm has the regret bound (4.3) (see [34]). Indeed, we will refer to $\boldsymbol{w}^{t+1} = \mathcal{P}(\dot{\boldsymbol{w}}^t; [\frac{\alpha}{n}, 1]^n \cap \Delta_n)$ as the "projection analogue" of (4.2). For tracking experts with memory our algorithm will instead project onto a set $\mathcal{C}$ that is updated on each trial, such that each weight does not fall below a certain threshold that is learned for each expert.

Given $\boldsymbol{\beta} \in (0, 1)^n$ such that $\|\boldsymbol{\beta}\|_1 \leq 1$, let

$$\mathcal{C}(\boldsymbol{\beta}) := \{\boldsymbol{x} \in \Delta_n : x_i \geq \beta_i, i = 1, \ldots, n\}$$

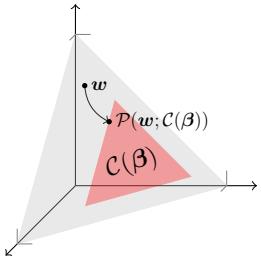


Figure 4.3: An illustration of the relative-entropy projection of the point $\boldsymbol{w} \in \Delta_3$ onto the set $\mathcal{C}(\boldsymbol{\beta}) = \{\boldsymbol{x} \in \Delta_3 : x_i \geq \beta_i, i = 1, \ldots, 3\}$, which is central to our algorithm.

be a subset of the simplex which is convex and non-empty. Given $\boldsymbol{w} \in \operatorname{ri} \Delta_n$, intuitively $\mathcal{P}(\boldsymbol{w};\mathcal{C}(\boldsymbol{\beta}))$ is the projection of $\boldsymbol{w}$ onto the simplex with (non-uniform) lower box constraints $\boldsymbol{\beta}$ (see Figure 4.3 for an illustration). Relative entropy projection updates for tracking experts with memory were first suggested in [1, Section 5.2]. The authors observed that for any MPP mixing scheme $\boldsymbol{\gamma}^{t+1}$, the update (4.5) can be replaced with

$$\boldsymbol{w}^{t+1} = \mathcal{P}(\dot{\boldsymbol{w}}^t; \{\boldsymbol{w} \in \Delta_n : \boldsymbol{w} \succeq \gamma_q^{t+1} \dot{\boldsymbol{w}}^q, q = 0, \ldots, t\}) , \tag{4.11}$$

and achieve the same regret bound. We build on this concept in this work. Observe that for any choice of $\boldsymbol{\gamma}^{t+1}$ the set $\{\boldsymbol{w} \in \Delta_n : \boldsymbol{w} \succeq \gamma_q^{t+1} \dot{\boldsymbol{w}}^q, q = 0, \ldots, t\}$ corresponds to the set $\mathcal{C}(\boldsymbol{\beta})$ where

$$\beta_i = \max_{0 \leq q \leq t} \gamma_q^{t+1} \dot{w}_i^q \qquad (i = 1, \ldots, n) . \tag{4.12}$$

In this work we give an algorithm to compute the projection $\mathcal{P}(\boldsymbol{w};\mathcal{C}(\boldsymbol{\beta}))$ exactly for any $\mathcal{C}(\boldsymbol{\beta})$ in $\mathcal{O}(n)$ time (Algorithm 5). With this algorithm and the mapping (4.12), one immediately obtains the projection analogue of MPP for any mixing scheme $\boldsymbol{\gamma}^{t+1}$ at essentially no additional computational cost. We point out, however, that for arbitrary mixing schemes computing $\boldsymbol{\beta}$ from (4.12) takes $\mathcal{O}(nt)$-time on trial t, improving only when some structure of the scheme can be exploited. We therefore propose the following method of **P**rojection **o**nto **D**ynamic **S**ets ("PoDS") for tracking experts with memory *efficiently*.

Algorithm 3 PoDS-θ

Input: $n > 0$, $\eta = \frac{1}{c} > 0$, $\alpha \in [0,1]$, $\theta \in [0,1]$

init: $\boldsymbol{w}^1 \leftarrow \frac{\mathbf{1}}{n}$; $\boldsymbol{\beta}^1 \leftarrow \alpha\frac{\mathbf{1}}{n}$
for $t \leftarrow 1$ to T **do**
  receive $\boldsymbol{x}^t \in \mathcal{D}^n$
  predict $\hat{y}^t = \texttt{pred}(\boldsymbol{w}^t, \boldsymbol{x}^t)$
  receive $y^t \in \mathcal{Y}$
  for $i \leftarrow 1$ to n **do**
    $\dot{w}_i^t \leftarrow \frac{w_i^t e^{-\eta \ell_i^t}}{\sum_{j=1}^n w_j^t e^{-\eta \ell_j^t}}$
  end for
  $\boldsymbol{w}^{t+1} \leftarrow \mathcal{P}(\dot{\boldsymbol{w}}^t; \mathcal{C}(\boldsymbol{\beta}^t))$ (4.13)
  $\boldsymbol{\beta}^{t+1} \leftarrow (1-\theta)\boldsymbol{\beta}^t + \theta\alpha\dot{\boldsymbol{w}}^t$
end for

Just as (4.8) generalizes the Fixed-Share update (4.2), we propose PoDS as the analogous generalization of the update $\boldsymbol{w}^{t+1} = \mathcal{P}(\dot{\boldsymbol{w}}^t; \mathcal{C}(\alpha\frac{\mathbf{1}}{n}))$ (the projection analogue of Fixed-Share). PoDS maintains a vector $\boldsymbol{\beta}^t \in \Delta_n^\alpha$, and on each trial updates the weights by setting $\boldsymbol{w}^{t+1} = \mathcal{P}(\dot{\boldsymbol{w}}^t; \mathcal{C}(\boldsymbol{\beta}^t))$. Intuitively PoDS is the projection analogue of (4.8) with $\boldsymbol{\beta}^t$ corresponding simply to $\alpha\boldsymbol{v}^t$. In some cases $\boldsymbol{\beta}^t = \alpha\boldsymbol{v}^t$ for all t (e.g., for Fixed-Share), but in general equality may not hold since $\boldsymbol{\beta}^t$ and $\boldsymbol{v}^t$ can be functions of past weights, which may differ for weight-sharing and projection algorithms. Recall that (4.8) encapsulates all MPP mixing schemes that set $\gamma_t^{t+1} = 1-\alpha$. PoDS implicitly captures the projection analogue of all such mixing schemes. This simple formulation of PoDS allows us to define new updates, which will correspond to new mixing schemes. In the following section we give a simple update for PoDS and improve on the best known regret bound. In Section 4.3.2 we discuss Algorithm 5 and the efficient computation of the projection $\mathcal{P}(\boldsymbol{w}; \mathcal{C}(\boldsymbol{\beta}))$.

4.3.1 A Simple Update Rule for PoDS

We now suggest a simple update rule for $\boldsymbol{\beta}^t$ in PoDS for tracking experts with memory. The regret bound for this algorithm is given in Theorem 15 (see Figure 4.2). We first set $\boldsymbol{\beta}^1 = \alpha\frac{\mathbf{1}}{n}$ to be uniform, and with a parameter

$0 \leq \theta \leq 1$ update $\boldsymbol{\beta}^t$ on subsequent trials by setting

$$\boldsymbol{\beta}^{t+1} = (1-\theta)\boldsymbol{\beta}^t + \theta\alpha\dot{\boldsymbol{w}}^t\,. \tag{4.14}$$

We refer to PoDS with this update as PoDS-θ. Intuitively the constraint vector $\boldsymbol{\beta}^t$ is updated in (4.14) by mixing in a small amount of the current weight vector, $\dot{\boldsymbol{w}}^t$, scaled such that $\|\boldsymbol{\beta}^{t+1}\|_1 = \alpha$. If expert i predicted well in the past, then its constraint β_i^t will be relatively large, preventing the weight from vanishing even if that expert suffers large losses locally. Using Algorithm 5 in its projection step, PoDS-θ has $\mathcal{O}(n)$ per-trial time complexity.

As discussed, the vector $\boldsymbol{\beta}^t$ of PoDS is conceptually equivalent to the vector $\alpha\boldsymbol{v}^t$ of the generalized share update (4.8). If PoDS has a simple update rule such as (4.14) then it is straightforward to recover the weight-sharing equivalent by simply "pretending" equality holds on all trials. We now do this for PoDS-θ. Clearly we have $\boldsymbol{v}^1 = \frac{\mathbf{1}}{n}$, and if $\boldsymbol{\beta}^t = \alpha\boldsymbol{v}^t$ and $\boldsymbol{\beta}^{t+1} = \alpha\boldsymbol{v}^{t+1}$, then $\boldsymbol{v}^{t+1} = \frac{1}{\alpha}\boldsymbol{\beta}^{t+1} = \frac{1}{\alpha}(1-\theta)\boldsymbol{\beta}^t + \theta\dot{\boldsymbol{w}}^t = (1-\theta)\boldsymbol{v}^t + \theta\dot{\boldsymbol{w}}^t$. This then leads to an efficient sharing algorithm, which we call Share-θ. In Section 4.5 we show this algorithm is in fact a new MPP mixing scheme, which surprisingly corresponds to the previous best known algorithm for this problem. Both PoDS-θ and Share-θ use the same parameters (α and θ), differing only in the final update (see Algorithms 3 and 4).

In the following theorem we give the regret bound which holds for both PoDS-θ and Share-θ.

Theorem 15. *For any comparison sequence $i_1, \ldots, i_T$ containing k switches and consisting of m unique experts from a set of size n, if $\alpha = \frac{k}{T-1}$ and $\theta = \frac{k-m+1}{(m-1)(T-2)}$, the regret of both PoDS-θ and Share-θ with any prediction function and loss function which are $(c, \frac{1}{c})$-realizable is*

$$\mathcal{R}(i_{1:T}) \leq c\left(m\ln n + (T-1)\mathcal{H}\left(\frac{k}{T-1}\right) + (m-1)(T-2)\mathcal{H}\left(\frac{k-m+1}{(m-1)(T-2)}\right)\right). \tag{4.17}$$

Algorithm 4 Share-θ

Input: $n > 0$, $\eta = \frac{1}{c} > 0$, $\alpha \in [0, 1]$, $\theta \in [0, 1]$

init: $\boldsymbol{w}^1 \leftarrow \frac{\mathbf{1}}{n}$; $\boldsymbol{v}^1 \leftarrow \frac{\mathbf{1}}{n}$
for $t \leftarrow 1$ to T **do**
  receive $\boldsymbol{x}^t \in \mathcal{D}^n$
  predict $\hat{y}^t = \texttt{pred}(\boldsymbol{w}^t, \boldsymbol{x}^t)$
  receive $y^t \in \mathcal{Y}$
  for $i \leftarrow 1$ to n **do**
    $\dot{w}_i^t \leftarrow \frac{w_i^t e^{-\eta \ell_i^t}}{\sum_{j=1}^n w_j^t e^{-\eta \ell_j^t}}$
  end for
  $\boldsymbol{w}^{t+1} \leftarrow (1 - \alpha)\dot{\boldsymbol{w}}^t + \alpha \boldsymbol{v}^t$ (4.15)
  $\boldsymbol{v}^{t+1} \leftarrow (1 - \theta)\boldsymbol{v}^t + \theta \dot{\boldsymbol{w}}^t$ (4.16)
end for

Proof. We first prove the bound for PoDS-θ, and then prove that Share-θ has the same bound. We use the relative entropy $D(\boldsymbol{u}^t, \boldsymbol{w}^t)$ as a measure of progress of the algorithm, where $\boldsymbol{u}^t$ is a comparator vector which we take to be a basis vector $\boldsymbol{e}_i$ for some $i \in [n]$ corresponding to the locally best expert i_t in hindsight on trial t. Recall that the comparator sequence $i_1, \ldots, i_T$ is partitioned with k *switches* into $k + 1$ *segments*, where a segment is defined as a sequence of trials where the comparator is unchanged, i.e. $i_a = \ldots = i_b$ for some $a < b$.

Recall that $\texttt{pred}$ and ℓ are assumed to be $(c, \frac{1}{c})$-realizable. That is, for any $\boldsymbol{w}^t \in \Delta_n$, $\boldsymbol{x}^t \in \mathcal{D}^n$, and $y^t \in \mathcal{Y}$, there exists $\eta > 0$ such that

$$\ell(\texttt{pred}(\boldsymbol{w}, \boldsymbol{x}), y) \leq -c \ln \sum_{i=1}^n w_i e^{-\eta \ell(x_i, y)} \tag{4.18}$$

holds with $c\eta = 1$.

We first establish that

$$\ell^t - \ell_{i_t}^t \leq c \left(D(\boldsymbol{u}^t, \boldsymbol{w}^t) - D(\boldsymbol{u}^t, \dot{\boldsymbol{w}}^t) \right) \tag{4.19}$$

holds for all t. Expanding the relative entropy terms gives

$$\begin{aligned} D(\boldsymbol{u}^t, \boldsymbol{w}^t) - D(\boldsymbol{u}^t, \dot{\boldsymbol{w}}^t) &= \sum_{i=1}^{n} u_i^t \ln \frac{\dot{w}_i^t}{w_i^t} \\ &= \sum_{i=1}^{n} u_i^t \ln \frac{w_i^t e^{-\eta \ell_i^t}}{w_i^t \sum_{j=1}^{n} w_j^t e^{-\eta \ell_j^t}} \\ &= -\eta \sum_{i=1}^{n} u_i^t \ell_i^t - \ln \sum_{j=1}^{n} w_j^t e^{-\eta \ell_j^t} \\ &\geq -\eta \ell_{i_t}^t + \frac{1}{c} \ell^t \,, \end{aligned}$$

where the inequality follows from (4.18). Multiplying both sides by c gives (4.19).

We now find lower bounds, δ, for $D(\boldsymbol{u}^t, \dot{\boldsymbol{w}}^t) - D(\boldsymbol{u}^{t+1}, \boldsymbol{w}^{t+1})$ to give non-negative terms of the form $D(\boldsymbol{u}^t, \dot{\boldsymbol{w}}^t) - D(\boldsymbol{u}^{t+1}, \boldsymbol{w}^{t+1}) - \delta \geq 0$, which we will multiply by c and add to (4.19) to give a telescoping sum of relative entropy terms. We consider three distinct cases for the different values of $\boldsymbol{u}^t$ over the T trials.

For the first case, we consider when there is no switch immediately after trial t (i.e., $\boldsymbol{u}^t = \boldsymbol{u}^{t+1}$). We use Corollary 19 with $\boldsymbol{u} = \boldsymbol{u}^t$, $\boldsymbol{w} = \dot{\boldsymbol{w}}^t$, and $\boldsymbol{\beta} = \boldsymbol{\beta}^t$. It follows then by definition that $\boldsymbol{p} = \boldsymbol{w}^{t+1}$ and we obtain

$$D(\boldsymbol{u}^t, \dot{\boldsymbol{w}}^t) - D(\boldsymbol{u}^{t+1}, \boldsymbol{w}^{t+1}) \geq \ln(1 - \alpha) \,, \tag{4.20}$$

which gives a telescoping sum of relative entropy terms within in each segment, paying $c \ln(1/(1-\alpha))$ for every trial where $\boldsymbol{u}^t = \boldsymbol{u}^{t+1}$.

For the two remaining cases, we will consider the segment boundaries, that is, the case when there is a switch and $\boldsymbol{u}^t \neq \boldsymbol{u}^{t+1}$. W.l.o.g let $\boldsymbol{u}^t = \boldsymbol{e}_j$ and let $\boldsymbol{u}^{t+1} = \boldsymbol{e}_k$ for any $j \neq k$ (that is we switch from expert "j" to expert "k" after

trial t). We then have the following

$$D(\boldsymbol{u}^t, \dot{\boldsymbol{w}}^t) - D(\boldsymbol{u}^{t+1}, \boldsymbol{w}^{t+1}) = \sum_{i=1}^{n} u_i^t \ln \frac{u_i^t}{\dot{w}_i^t} - \sum_{i=1}^{n} u_i^{t+1} \ln \frac{u_i^{t+1}}{w_i^{t+1}}$$
$$= \ln \frac{1}{\dot{w}_j^t} + \ln w_k^{t+1}\,, \tag{4.21}$$

thus we collect a $\ln(1/\dot{w}_j^t)$ term from the *last* trial of the segment of expert j and a $\ln(w_k^{t+1})$ term from the *first* trial of the new segment of expert k. We now consider the remaining two cases: when trial $t+1$ is the first time expert k predicts well, and when trial $t+1$ is a trial on which we "re-visit" expert k.

For the first of these two cases, we consider the first time expert k starts to predict well. We then use (4.13) and (4.14) to give

$$\ln w_k^{t+1} \geq \ln \beta_k^t \geq \ln\left((1-\theta)^{t-1}\beta_k^1\right) = \ln\left((1-\theta)^{t-1}\frac{\alpha}{n}\right). \tag{4.22}$$

Substituting (4.22) into (4.21), we therefore pay $-c\ln((1-\theta)^{t-1}\frac{\alpha}{n})$ to switch to a new expert for the first time on trial $t+1$.

Finally for the second of these two cases, we consider when expert k has predicted well before. Let trial $q < t$ denote the *last* trial of expert k's most recent "segment." We then have the following (again using (4.13) and (4.14)),

$$\ln w_k^{t+1} \geq \ln \beta_k^t \geq \ln\left((1-\theta)^{t-q-1}\beta_k^{q+1}\right) \geq \ln\left((1-\theta)^{t-q-1}\alpha\theta\dot{w}_k^q\right). \tag{4.23}$$

By substituting (4.23) into (4.21) for each segment boundary, and summing over these boundaries, we therefore pay $-c\ln((1-\theta)^{t-q-1}\alpha\theta)$ in order to telescope the $\ln(\dot{w}_k^q)$ term with the $\ln(1/\dot{w}_k^q)$ term from the end of expert k's most recent segment ending on trial q.

Putting these together we thus pay $c\ln(1/(1-\alpha))$ for every trial on which we don't switch (from Corollary 19), we pay $c\ln(1/(1-\theta))$ for every expert in our pool that *isn't* predicting well or involved in a switch on every trial (i.e., $m-1$ times, on non-switch trials, and $m-2$ times on switch trials, from (4.22) and (4.23)), and finally when we switch to an expert k before trial $t+1$ we pay

$c \ln (n/\alpha)$ if it is the first time to track expert k (there are $m-1$ such trials), and $c \ln (1/\alpha\theta)$ otherwise (there are $k-m+1$ such trials).

Summing over all trials, and using $D(\boldsymbol{u}^1, \boldsymbol{w}^1) \leq \ln n$ then gives

$$\begin{aligned}
\sum_{t=1}^{T} \ell^t - \sum_{t=1}^{T} \ell_{i_t}^t &\leq \sum_{t=1}^{T} c \left(D(\boldsymbol{u}^t, \boldsymbol{w}^t) - D(\boldsymbol{u}^t, \dot{\boldsymbol{w}}^t) + D(\boldsymbol{u}^t, \dot{\boldsymbol{w}}^t) - D(\boldsymbol{u}^{t+1}, \boldsymbol{w}^{t+1}) \right) \\
&\leq cD(\boldsymbol{u}^1, \boldsymbol{w}^1) + c(T-k-1) \ln \left(\frac{1}{1-\alpha} \right) + c(m-1) \ln \left(\frac{n}{\alpha} \right) \\
&\quad + c((m-1)(T-1)-k) \ln \left(\frac{1}{1-\theta} \right) \\
&\quad + c(k-m+1) \ln \left(\frac{1}{\alpha\theta} \right) \\
&\leq cm \ln n + c(T-k-1) \ln \left(\frac{1}{1-\alpha} \right) + ck \ln \left(\frac{1}{\alpha} \right) \\
&\quad + c((m-1)(T-1)-k) \ln \left(\frac{1}{1-\theta} \right) \\
&\quad + c(k-m+1) \ln \left(\frac{1}{\theta} \right). \qquad (4.24)
\end{aligned}$$

The optimal tuning of α and θ that minimizes (4.24) is given by $\alpha = \frac{k}{T-1}$ and $\theta = \frac{k-m+1}{(m-1)(T-2)}$. Substituting these values into (4.24) gives a bound of

$$cm \ln n + c(T-1)\mathcal{H}\left(\frac{k}{T-1} \right) + c(m-1)(T-2)\mathcal{H}\left(\frac{k-m+1}{(m-1)(T-2)} \right),$$

which completes the proof for PoDS-θ.

We now prove that Share-θ has the same bound with an almost identical argument as the proof just given for PoDS-θ. Firstly observe that (4.21) is independent of the algorithm update and therefore holds for both algorithms. Additionally, observe that the proof for PoDS-θ relies on the inequalities (4.19), (4.20), (4.22), and (4.23). We now prove that these inequalities hold for Share-θ, and thus the two algorithms share the same bound.

Firstly we observe that inequality (4.19) holds since both algorithms use the same loss update, and we assume that the prediction function and loss function are $(c, \frac{1}{c})$-realizable.

Secondly, it follows directly from the update (4.15) that (4.20) holds for Share-θ when $\boldsymbol{u}^t = \boldsymbol{u}^{t+1}$, since $\boldsymbol{w}^{t+1} \geq (1-\alpha)\dot{\boldsymbol{w}}^t$ and therefore

$$D(\boldsymbol{u}^t, \dot{\boldsymbol{w}}^t) - D(\boldsymbol{u}^{t+1}, \boldsymbol{w}^{t+1}) = \sum_{i=1}^{n} u_i^t \ln \frac{w_i^{t+1}}{\dot{w}_i^t} \geq \sum_{i=1}^{n} u_i^t \ln \frac{(1-\alpha)\dot{w}_i^t}{\dot{w}_i^t} = \ln(1-\alpha)\,.$$

The proof that (4.22) holds follows directly from the updates (4.15) and (4.16) and the fact $\boldsymbol{v}^1 = \frac{\boldsymbol{1}}{n}$. That is, for the first time expert "k" appears on trial $t+1$,

$$\ln w_k^{t+1} \geq \ln(\alpha v_k^t) \geq \ln\left((1-\theta)^{t-1}\alpha v_k^1\right) = \ln\left((1-\theta)^{t-1}\frac{\alpha}{n}\right).$$

Similarly, the proof that (4.23) holds follows directly from the updates (4.15) and (4.16). That is, when we return to expert "k" on trial $t+1$,

$$\ln w_k^{t+1} \geq \ln(\alpha v_k^t) \geq \ln\left((1-\theta)^{t-q-1}\alpha v_k^{q+1}\right) \geq \ln\left((1-\theta)^{t-q-1}\alpha\theta \dot{w}_k^q\right).$$

Having shown that the inequalities (4.19), (4.20), (4.22), and (4.23) hold for Share-θ, the remainder of the proof follows exactly as the proof for PoDS-θ. □

The regret bound (4.17) is at least $c((m-1)\ln\frac{T-1}{k} - (k-m+1)\ln\frac{k}{k-m+1})$ tighter than the currently best known bound (4.9). Thus if $m \ll k$ then the improvement is $\approx cm\ln\frac{T}{k}$, and as $m \to k+1$ then the improvement is $\approx ck\ln\frac{T}{k}$. Additionally note that if $m = k+1$ (i.e., every switch we track a *new* expert) the optimal tuning of θ is zero, and PoDS-θ reduces to setting $\boldsymbol{\beta}^t = \alpha\frac{\boldsymbol{1}}{n}$ on every trial. That is, we recover the projection analogue of Fixed-Share. This is also reflected in the regret bound since (4.17) reduces to (4.3). Since $x\mathcal{H}(\frac{y}{x}) \leq y\ln(\frac{x}{y}) + y$, the regret bound (4.17) is upper-bounded by

$$\mathcal{R}(i_{1:T}) \leq c\Bigg[m\ln n + k\ln\frac{T-1}{k} + (k-m+1)\ln\frac{T-2}{k-m+1} + (k-m+1)\ln(m-1) + 2k - m + 1\Bigg].$$

Comparing this to (4.10), we see that instead of paying $c\ln\frac{T-1}{k}$ *twice* on every

Algorithm 5 $\mathcal{P}(\boldsymbol{w};\mathcal{C}(\boldsymbol{\beta}))$ in $\mathcal{O}(n)$ time

Input: $\boldsymbol{w} \in \text{ri } \Delta_n; \boldsymbol{\beta} \in (0,1)^n$ s.t. $\|\boldsymbol{\beta}\|_1 \leq 1$
Output: $\boldsymbol{w}' = \mathcal{P}(\boldsymbol{w};\mathcal{C}(\boldsymbol{\beta}))$

init: $\mathcal{W} \leftarrow [n]$; $\boldsymbol{r} \leftarrow \boldsymbol{w} \odot \frac{1}{\boldsymbol{\beta}}$; $S_{\boldsymbol{w}} \leftarrow 0$; $S_{\boldsymbol{\beta}} \leftarrow 0$
while $\mathcal{W} \neq \emptyset$ **do**
  $\phi \leftarrow \texttt{median}(\{r_i : i \in \mathcal{W}\})$
  $\mathcal{L} \leftarrow \{i \in \mathcal{W} : r_i < \phi\}$
  $L_{\boldsymbol{\beta}} \leftarrow \sum_{i\in\mathcal{L}} \beta_i$; $L_{\boldsymbol{w}} \leftarrow \sum_{i\in\mathcal{L}} w_i$
  $\mathcal{M} \leftarrow \{i \in \mathcal{W} : r_i = \phi\}$
  $M_{\boldsymbol{\beta}} \leftarrow \sum_{i\in\mathcal{M}} \beta_i$; $M_{\boldsymbol{w}} \leftarrow \sum_{i\in\mathcal{M}} w_i$
  $\mathcal{H} \leftarrow \{i \in \mathcal{W} : r_i > \phi\}$
  $\lambda \leftarrow \frac{1-S_{\boldsymbol{\beta}}-L_{\boldsymbol{\beta}}}{1-S_{\boldsymbol{w}}-L_{\boldsymbol{w}}}$
  if $\phi\lambda < 1$ **then**
    $S_{\boldsymbol{w}} \leftarrow S_{\boldsymbol{w}} + L_{\boldsymbol{w}} + M_{\boldsymbol{w}}$
    $S_{\boldsymbol{\beta}} \leftarrow S_{\boldsymbol{\beta}} + L_{\boldsymbol{\beta}} + M_{\boldsymbol{\beta}}$
    if $\mathcal{H} = \emptyset$ **then**
      $\phi \leftarrow \texttt{min}(\{r_i : r_i > \phi, i \in [n]\})$
    end if
    $\mathcal{W} \leftarrow \mathcal{H}$
  else
    $\mathcal{W} \leftarrow \mathcal{L}$
  end if
end while
$\lambda \leftarrow \frac{1-S_{\boldsymbol{\beta}}}{1-S_{\boldsymbol{w}}}$
$\forall i : 1, \ldots, n : w'_i \leftarrow \begin{cases} \beta_i & r_i < \phi \\ \lambda w_i & r_i \geq \phi \end{cases}$

switch, we pay $c \ln \frac{T-1}{k}$ once per switch and $c \ln \frac{T-2}{k-m+1}$ for every switch we *remember* an old expert ($k - m + 1$ times). Unlike previous results for tracking experts with memory, PoDS-θ and its regret bound (4.17) smoothly interpolate between the two switching settings. That is, it is capable of exploiting memory when necessary and on the other hand does not suffer when memory is not necessary (see Figure 4.2).

4.3.2 Computing $\mathcal{P}(\boldsymbol{w};\mathcal{C}(\boldsymbol{\beta}))$

Before we consider PoDS-θ and Share-θ further, we discuss the computation of the projection $\mathcal{P}(\boldsymbol{w};\mathcal{C}(\boldsymbol{\beta}))$. In [37] the authors showed that computing relative entropy projection onto the simplex with *uniform* box constraints is non-trivial, but gave an algorithm to compute it in $\mathcal{O}(n)$ time. We give a generalization

of their algorithm to compute $\mathcal{P}(\boldsymbol{w};\mathcal{C}(\boldsymbol{\beta}))$ exactly for any non-empty set $\mathcal{C}(\boldsymbol{\beta})$ in $\mathcal{O}(n)$ time. As far as we are aware our method to compute exact relative entropy projection onto the simplex with non-uniform (lower) box constraints in linear time is the first, and may be of independent interest (see e.g., [80]).

We first develop intuition by sketching out the form that $\mathcal{P}(\boldsymbol{w};\mathcal{C}(\boldsymbol{\beta}))$ must take, and then describe how Algorithm 5 computes this projection efficiently. This is stated formally in Theorem 16. Firstly consider the case that $\boldsymbol{w} \in \mathcal{C}(\boldsymbol{\beta})$, then trivially $\mathcal{P}(\boldsymbol{w};\mathcal{C}(\boldsymbol{\beta})) = \boldsymbol{w}$, due to the non-negativity of $D(\boldsymbol{u},\boldsymbol{w})$ and the fact that $D(\boldsymbol{u},\boldsymbol{w}) = 0$ if and only if $\boldsymbol{u} = \boldsymbol{w}$ [79]. For the case that $\boldsymbol{w} \notin \mathcal{C}(\boldsymbol{\beta})$, this implies that the set $\{i \in [n] : w_i < \beta_i\}$ is non-empty. For each index i in this set, we will see that the projection of $\boldsymbol{w}$ onto $\mathcal{C}(\boldsymbol{\beta})$ must set the component w_i to its corresponding constraint value β_i. The remaining components are then normalized, such that $\sum_{i=1}^{n} w_i = 1$. However, doing so may cause one (or more) of these components w_j to drop below its constraint β_j. In the proof of Theorem 16 we show that the projection algorithm must find the set of components Ψ of least cardinality to set to their constraint values such that when the remaining components are normalized, no component lies below its constraint, and that this can be done in linear time.

Consider the following inefficient approach to finding Ψ. Given $\boldsymbol{w}$ and $\mathcal{C}(\boldsymbol{\beta})$, let $\boldsymbol{r} = \boldsymbol{w} \odot \frac{1}{\boldsymbol{\beta}}$ be a "ratio vector." First sort $\boldsymbol{r}$ in ascending order, and then sort $\boldsymbol{w}$ and $\boldsymbol{\beta}$ according to the ordering of $\boldsymbol{r}$. If $r_1 \geq 1$ then $\Psi = \emptyset$ and we are done ($\Rightarrow \boldsymbol{w} \in \mathcal{C}(\boldsymbol{\beta})$). Otherwise for each $a = 1, \ldots, n$:

1. Let the candidate set $\Psi' = [a]$.
2. Let $\boldsymbol{w}' = \boldsymbol{w}$ except for each $i \in \Psi'$ set $w'_i = \beta_i$.
3. Re-normalize the remaining components of $\boldsymbol{w}'$.
4. Let $\boldsymbol{r}' = \boldsymbol{w}' \odot \frac{1}{\boldsymbol{\beta}}$.

The set Ψ is then the candidate set Ψ' of least cardinality such that $\boldsymbol{r}' \succeq \mathbf{1}$. This approach requires sorting $\boldsymbol{r}$ and therefore even an efficient implementation

takes $\mathcal{O}(n \log n)$ time. Algorithm 5 finds Ψ without having to sort $\boldsymbol{r}$. It instead specifies Ψ uniquely with a threshold, ϕ, such that $\Psi = \{i : r_i < \phi\}$. Algorithm 5 finds ϕ through repeatedly bisecting the set $\mathcal{W} = [n]$ by finding the median of the set $\{r_i : i \in \mathcal{W}\}$ (which can be done in $\mathcal{O}(|\mathcal{W}|)$ time [81]), and efficiently testing this value as the candidate threshold on each iteration. The smallest valid threshold then specifies the set Ψ. The following theorem states the time complexity of the algorithm and the form of the projection, which is used in the proof of Theorem 15.

Theorem 16. *For any $\boldsymbol{\beta} \in (0,1)^n$ such that $\|\boldsymbol{\beta}\|_1 \leq 1$, and for any $\boldsymbol{w} \in \text{ri } \Delta_n$, let $\boldsymbol{p} = \mathcal{P}(\boldsymbol{w}; \mathcal{C}(\boldsymbol{\beta}))$, where $\mathcal{C}(\boldsymbol{\beta}) = \{\boldsymbol{x} \in \Delta_n : x_i \geq \beta_i, i = 1, \ldots, n\}$. Then $\boldsymbol{p}$ is such that for all $i = 1, \ldots, n$,*

$$p_i = \max \left\{ \beta_i; \frac{1 - \sum_{j \in \Psi} \beta_j}{1 - \sum_{j \in \Psi} w_j} w_i \right\}, \tag{4.25}$$

where $\Psi := \{i \in [n] : p_i = \beta_i\}$. Furthermore, Algorithm 5 computes $\boldsymbol{p}$ in $\mathcal{O}(n)$ time.

We now prove the two statements of Theorem 16 separately. The proof of the theorem follows very closely to the proof of Theorem 7 in [37] (including Claims 1, 2, and 3). There the problem is concerned with uniform constraints, whereas we consider non-uniform constraints. In particular Claims 17 and 18 given below are generalizations of Claims 2 and 3 of [37]. The proof of the second statement of Theorem 16 is almost identical to the proof of Theorem 7 in [37]. We first give a sketch of the proofs of the two statements of Theorem 16.

For the first statement, recall that $\Psi := \{i \in [n] : p_i = \beta_i\}$ is the set of indexes of components which must be set to their constraint values. To prove the first statement we will show that given $\boldsymbol{w}$ and $\mathcal{C}(\boldsymbol{\beta})$, each component of the point $\mathcal{P}(\boldsymbol{w}; \mathcal{C}(\boldsymbol{\beta}))$ either takes the value of its lower box constraint, β_i, or is equal to w_i multiplied by a factor λ, with

$$\lambda = \frac{1 - \sum_{i \in \Psi} \beta_i}{1 - \sum_{i \in \Psi} w_i}.$$

We then argue that each component $p_i = \max\{\beta_i; \lambda w_i\}$ for $i = 1, \ldots, n$.

For the second statement, we first show that Ψ, which uniquely specifies $\mathcal{P}(\boldsymbol{w}; \mathcal{C}(\boldsymbol{\beta}))$, is the set of minimum cardinality such that when all other components are re-normalized, no component lies below its constraint value, and then show that Algorithm 5 finds this set in $\mathcal{O}(n)$ time.

Proof of the first statement of Theorem 16. Recall the first statement of the theorem: that $\mathcal{P}(\boldsymbol{w}; \mathcal{C}(\boldsymbol{\beta}))$ takes the form (4.25). Given $\boldsymbol{w}$ and the non-empty set $\mathcal{C}(\boldsymbol{\beta})$, the point $\mathcal{P}(\boldsymbol{w}; \mathcal{C}(\boldsymbol{\beta}))$ is the minimizer of the following convex optimization problem

$$\begin{aligned} \min_{\boldsymbol{u}} \quad & D(\boldsymbol{u}, \boldsymbol{w}) \\ \text{s.t.} \quad & \beta_i - u_i \leq 0, \quad i = 1, \ldots, n \\ & \mathbf{1} \cdot \boldsymbol{u} - 1 = 0 \quad . \end{aligned} \tag{4.26}$$

Since $D(\boldsymbol{u}, \boldsymbol{w})$ is convex in its first argument, and $\mathcal{C}(\boldsymbol{\beta})$ is a convex set, then (4.26) has a unique minimizer, which we denote by $\boldsymbol{p}$.

Constructing the Lagrangian of (4.26) with Lagrange multipliers $\boldsymbol{\xi} \succeq \mathbf{0}, \nu \in \mathbb{R}$,

$$\mathcal{L}(\boldsymbol{u}, \boldsymbol{\xi}, \nu) = \sum_{i=1}^{n} u_i \ln \frac{u_i}{w_i} + \boldsymbol{\xi}^\top (\boldsymbol{\beta} - \boldsymbol{u}) + \nu(\mathbf{1} \cdot \boldsymbol{u} - 1)\,,$$

and setting $\nabla_{\boldsymbol{u}} \mathcal{L}(\boldsymbol{u}, \boldsymbol{\xi}, \nu) = \mathbf{0}$ gives for $i = 1, \ldots, n$,

$$\frac{\partial \mathcal{L}}{\partial u_i} = \ln \frac{u_i}{w_i} + 1 - \xi_i + \nu = 0\,.$$

This then gives for $i = 1, \ldots, n$,

$$p_i = w_i e^{\xi_i - 1 - \nu}\,.$$

Since $D(\boldsymbol{u}, \boldsymbol{w})$ is convex in its first argument, and (4.26) has only linear constraints then strong duality holds and we may exploit the complementary

slackness Karush-Kuhn-Tucker necessary condition of the optimal solution (see e.g., [82, Chapter 5]). That is, $\xi_i(\beta_i - p_i) = 0$ for all $i = 1, \ldots, n$. Therefore for any i such that $p_i > \beta_i$, the corresponding Lagrange multiplier is zero, and we have

$$p_i = w_i e^{-1-\nu} .$$

Recall $\Psi = \{i : p_i = \beta_i\}$, we then have

$$1 = \sum_{i=1}^{n} p_i = \sum_{i\in\Psi} p_i + \sum_{i\in[n]\setminus\Psi} p_i = \sum_{i\in\Psi} \beta_i + \sum_{i\in[n]\setminus\Psi} w_i e^{-1-\nu} .$$

Re-arranging gives

$$e^{-1-\nu} = \frac{1 - \sum_{i\in\Psi} \beta_i}{\sum_{i\in[n]\setminus\Psi} w_i} = \frac{1 - \sum_{i\in\Psi} \beta_i}{1 - \sum_{i\in\Psi} w_i} .$$

Therefore for each index $i \in [n]$, either i is in Ψ which implies $p_i = \beta_i$, or $i \notin \Psi$ and therefore $p_i = \lambda w_i$, where

$$\lambda = \frac{1 - \sum_{j\in\Psi} \beta_j}{1 - \sum_{j\in\Psi} w_j} .$$

We now establish that $p_i = \max\{\beta_i; \lambda w_i\}$ for all $i = 1, \ldots, n$. Observe that if $i \in \Psi$, then $p_i = w_i e^{\xi_i - 1 - \nu} = \beta_i$, and since the Lagrange multiplier $\xi_i \geq 0$ then $p_i \geq w_i e^{-1-\nu} = \lambda w_i$.

For $i \notin \Psi$, then this implies $p_i = \lambda w_i > \beta_i$, since if $p_i = \beta_i$ then $i \in \Psi$, and if $p_i < \beta_i$ then we have a contradiction since $\boldsymbol{p}$ is not a feasible solution to (4.26). We therefore conclude that $\boldsymbol{p}$ is such that for all $i = 1, \ldots, n$,

$$p_i = \max\left\{\beta_i; \frac{1 - \sum_{j\in\Psi} \beta_j}{1 - \sum_{j\in\Psi} w_j} w_i\right\} ,$$

which completes the proof of the first statement of the Theorem. □

The proof of the second statement of Theorem 16 will rely on the following two claims.

Claim 17. *Given* $\boldsymbol{w}$ *and* $\boldsymbol{\beta}$*, let* $\boldsymbol{r} := \boldsymbol{w} \odot \frac{1}{\boldsymbol{\beta}}$*. Without loss of generality, for* $i < j$ *assume* $r_i \leq r_j$*. Let* $\lambda = \frac{1-\sum_{i\in\Psi}\beta_i}{1-\sum_{i\in\Psi} w_i}$*, then*

$$\boldsymbol{p} = \left(\beta_1, \ldots, \beta_{|\Psi|}, \lambda w_{|\Psi|+1}, \ldots, \lambda w_n\right) . \tag{4.27}$$

Proof. In the proof of the first statement of Theorem 16 we established that $\boldsymbol{p}$ is a permutation of (4.27), that is, either $p_i = \beta_i$ or $p_i = \lambda w_i$ for $i = 1, \ldots, n$. We also established that $p_i = \max\{\beta_i; \lambda w_i\}$ for $i = 1, \ldots, n$.

Suppose $\boldsymbol{p}$ is not in the form of (4.27). Then there exists $a < b$ such that $p_a = \lambda w_a$ and $p_b = \beta_b$ (that is, $b \in \Psi$ and $a \notin \Psi$).

If $p_a = \lambda w_a$ then by the first statement of Theorem 16 we have $\lambda w_a > \beta_a$. However since $r_a \leq r_b$, and $\lambda > 0$, this implies $\frac{\lambda w_a}{\beta_a} \leq \frac{\lambda w_b}{\beta_b}$. We then have $1 < \frac{\lambda w_a}{\beta_a} \leq \frac{\lambda w_b}{\beta_b}$, which implies $\lambda w_b > \beta_b$. However we necessarily assumed that $p_b = \beta_b$. This violates the first statement of Theorem 16 that $p_b = \max\{\lambda w_b, \beta_b\}$, and thus contradicts our assumption that $\boldsymbol{p}$ is the minimizer of (4.26). Hence our supposition that $\boldsymbol{p}$ is not in the form of (4.27) is false. □

Claim 18. *Let* $\Psi' = \{1, \ldots, k\}$*, and* $\Psi'' = \{1, \ldots, k{+}1\}$*, and let* $\lambda' = \frac{1-\sum_{i\in\Psi'}\beta_i}{1-\sum_{i\in\Psi'} w_i}$*, and* $\lambda'' = \frac{1-\sum_{i\in\Psi''}\beta_i}{1-\sum_{i\in\Psi''} w_i}$*. Then let*

$$\boldsymbol{u}' = \left(\overbrace{\beta_1, \ldots, \beta_{|\Psi'|}}^{k}, \lambda' w_{|\Psi'|+1}, \ldots, \lambda' w_n\right) ,$$

and

$$\boldsymbol{u}'' = \left(\overbrace{\beta_1, \ldots, \beta_{|\Psi''|}}^{k+1}, \lambda'' w_{|\Psi''|+1}, \ldots, \lambda'' w_n\right) ,$$

then $D(\boldsymbol{u}', \boldsymbol{w}) \leq D(\boldsymbol{u}'', \boldsymbol{w})$*.*

Proof. Consider the following convex optimization problem for some $\boldsymbol{w} \in \operatorname{ri} \Delta_n$,

$$\begin{aligned} \min_{\boldsymbol{u}} \quad & D(\boldsymbol{u}, \boldsymbol{w}) \\ \text{s.t.} \quad & \beta_i - u_i = 0, \quad i = 1, \ldots, k \\ & \mathbf{1} \cdot \boldsymbol{u} - 1 = 0 \quad . \end{aligned} \tag{4.28}$$

The point $\boldsymbol{u}'$ is the unique minimizer of (4.28), while $\boldsymbol{u}''$ clearly also satisifies the constraints of (4.28) and is therefore a feasible solution. This implies that $D(\boldsymbol{u}', \boldsymbol{w}) \leq D(\boldsymbol{u}'', \boldsymbol{w})$. □

Proof of the second statement of Theorem 16. Recall the second statement of the theorem: that Algorithm 5 computes $\mathcal{P}(\boldsymbol{w}; \mathcal{C}(\boldsymbol{\beta}))$ in linear time. We prove this statement by first showing that the set Ψ corresponding to this projection is the set of components of minimal cardinality to set to their constraint values such that when the other components are normalized, no component lies below its constraint value. We then prove that Algorithm 5 computes the projection by finding this set in linear time.

In the proof of the first statement of the theorem we proved that $\boldsymbol{p}$ has the form (4.25). Thus $\boldsymbol{p}$ is uniquely specified by the set $\Psi = \{i \in [n] : p_i = \beta_i\} \subseteq \{1, \ldots, n\}$. There are therefore 2^n possible solutions. Claim 17 proves that the magnitude of the ratio of a component and its constraint is smaller for a component to be set to its constraint value than a component to be normalized. That is, if $i \in \Psi$ and $j \notin \Psi$, then $\frac{w_i}{\beta_i} \leq \frac{w_j}{\beta_j}$. This reduces the number of feasible solutions to n.

Given these n possible solutions, claim 18 shows that if $\Psi' \subseteq \Psi''$ with corresponding candidate projection vectors $\boldsymbol{u}'$ and $\boldsymbol{u}''$ respectively, then $D(\boldsymbol{u}', \boldsymbol{w}) \leq D(\boldsymbol{u}'', \boldsymbol{w})$. Thus to compute the projection, one must find the set Ψ of minimum cardinality whose corresponding candidate projection vector is in $\mathcal{C}(\boldsymbol{\beta})$.

Observe that this "minimal" set Ψ is specified uniquely by a threshold, ϕ, such that $\Psi = \{i \in [n] : r_i < \phi\}$, where $r_i = \frac{w_i}{\beta_i}$, for $i = 1, \ldots, n$. Algorithm 5 finds Ψ by finding this threshold. The algorithm initially computes the vector $\boldsymbol{r} = \boldsymbol{w} \odot \frac{1}{\boldsymbol{\beta}}$ and when ϕ has been found, the algorithm sets all components of w_i where $r_i < \phi$ to their thresholds β_i, and normalizes the remaining components.

We now discuss how the algorithm finds ϕ in linear time. On each iteration a candidate threshold is examined. These candidate thresholds are determined from an index set $\mathcal{W}$, which is initially set to $\{1, \ldots, n\}$. On each iteration

the threshold ϕ is chosen as the median of the ratios in the set $\{r_i : i \in \mathcal{W}\}$ (line 3). This can be done in $\mathcal{O}(|\mathcal{W}|)$ time [81]. The approach used is a divide and conquer method, however from a practical perspective this could also be replaced with a randomized median-finding algorithm with average time complexity $\mathcal{O}(|\mathcal{W}|)$ [83]. If $|\mathcal{W}|$ is even, then the algorithm can choose between the $\frac{|\mathcal{W}|}{2}$ and the $\frac{|\mathcal{W}|+1}{2}$ largest element arbitrarily. The set $\mathcal{W}$ is then sorted into two sets, $\mathcal{L}$ and $\mathcal{H}$, where $\mathcal{L} = \{i \in \mathcal{W} : r_i < \phi\}$ and $\mathcal{H} = \{i \in \mathcal{W} : r_i > \phi\}$.

The normalizing constant λ is then computed (line 9). If $\lambda\phi < 1$, then by Claims 17 and 18 the true threshold must be larger than the current candidate threshold ϕ, and must therefore correspond to r_i for an index i contained in $\mathcal{H}$. Otherwise the true threshold must be either equal to the current candidate threshold, or must correspond to r_i for an index i contained in $\mathcal{L}$.

Since ϕ was taken to be the median, then the algorithm iterates this procedure, setting $\mathcal{W} = \mathcal{L}$ or $\mathcal{W} = \mathcal{H}$ as appropriate. Additionally, since ϕ was taken to be the median, then $\max\{|\mathcal{L}|; |\mathcal{H}|\} \leq \frac{1}{2}|\mathcal{W}|$. When $\mathcal{W} = \emptyset$, then the algorithm has found ϕ, and the projection is computed.

There are a maximum of $\lceil \log n + 1 \rceil$ iterations of lines 2-18, with the i^{th} iteration taking $\mathcal{O}(\frac{n}{2^i})$ time. The algorithm therefore takes $\mathcal{O}(n)$ time to find ϕ, and the time complexity of the algorithm is therefore $\mathcal{O}(n)$. □

We have the following Corollary which follows from Theorem 16, which we use in our proof of Theorem 15.

Corollary 19. *Let* $0 < \alpha < 1$. *Then for any* $\boldsymbol{u} \in \Delta_n$, $\boldsymbol{w} \in \text{ri } \Delta_n$, *and* $\boldsymbol{\beta} \in \text{ri } \Delta_n^\alpha$, *let* $\boldsymbol{p} = \mathcal{P}(\boldsymbol{w}; \mathcal{C}(\boldsymbol{\beta}))$. *Then,*

$$D(\boldsymbol{u}, \boldsymbol{w}) - D(\boldsymbol{u}, \boldsymbol{p}) \geq \ln(1 - \alpha). \tag{4.29}$$

Proof. Let $\Psi := \{i \in [n] : p_i = \beta_i\}$. Recall from Theorem 16 that the projected vector $\boldsymbol{p}$ takes the form (4.25). Expanding the relative entropy terms of (4.29)

then gives the following,

$$\begin{aligned} D(\boldsymbol{u}, \boldsymbol{w}) - D(\boldsymbol{u}, \boldsymbol{p}) &= \sum_{i=1}^{n} u_i \ln\left(\frac{p_i}{w_i}\right) \\ &\geq \sum_{i=1}^{n} u_i \ln\left(\frac{\left(1 - \sum_{j\in\Psi} \beta_j\right) w_i}{\left(1 - \sum_{j\in\Psi} w_j\right) w_i}\right) \\ &= \ln\left(\frac{1 - \sum_{j\in\Psi} \beta_j}{1 - \sum_{j\in\Psi} w_j}\right) \\ &\geq \ln\left(1 - \alpha\right), \end{aligned}$$

where the first inequality follows from the definition of p_i in (4.25) and the fact that $\max\{a, b\} \geq b$. The second inequality follows from the fact that $\sum_{j\in\Psi} w_j \geq 0$ and $\sum_{j\in\Psi} \beta_j \leq \alpha$. □

4.4 Projection vs. Sharing in Online Learning

We now briefly consider the two types of updates discussed in this chapter (projection and weight-sharing) when updating weights may incur costs. Recall the motivating example introduced in Section 4.2 was in online portfolio selection with transaction costs. It is straightforward to show that in this model transaction costs are proportional to the ℓ_1-norm of the difference in the weight vectors before and after re-balancing, that is, $\|\boldsymbol{w}^{t+1} - \dot{\boldsymbol{w}}^t\|_1$. In Theorem 20 we give a result which in this context guarantees the "cost" of projecting is less than that of weight-sharing.

To compare the update of PoDS and the generalized share update (4.8), we must consider for a set of weights $\dot{\boldsymbol{w}}^t$, the point $\mathcal{P}(\dot{\boldsymbol{w}}^t; \mathcal{C}(\boldsymbol{\beta}^t))$ and the point $(1-\alpha)\dot{\boldsymbol{w}}^t + \alpha\boldsymbol{v}^t$. However these points depend on $\boldsymbol{\beta}^t$ and $\boldsymbol{v}^t$ respectively, which may themselves be functions of previous weight vectors $\dot{\boldsymbol{w}}^1, \ldots, \dot{\boldsymbol{w}}^{t-1}$, which as discussed are generally not the same for each of the two algorithms. To compare the two updates equally we therefore assume that the current weights are the same (i.e., they must both update the same weights $\dot{\boldsymbol{w}}^t$), and additionally that $\boldsymbol{\beta}^t = \alpha\boldsymbol{v}^t$. The following theorem states that under mild conditions, PoDS is

strictly less "expensive" than its weight-sharing counterpart.

Theorem 20. *Let $0 < \alpha < 1$. Then for any $\boldsymbol{v} \in \text{ri } \Delta_n$, let $\boldsymbol{\beta} = \alpha\boldsymbol{v}$, and for any $\boldsymbol{w} \in \text{ri } \Delta_n$ such that $\boldsymbol{w} \neq \boldsymbol{v}$, let $S(\boldsymbol{w}, \boldsymbol{v}) := (1-\alpha)\boldsymbol{w} + \alpha\boldsymbol{v}$. Then,*

$$\|\mathcal{P}(\boldsymbol{w}; \mathcal{C}(\boldsymbol{\beta})) - \boldsymbol{w}\|_1 < \|S(\boldsymbol{w}, \boldsymbol{v}) - \boldsymbol{w}\|_1 \ .$$

Before proving Theorem 20, we introduce some additional notation. Let $\boldsymbol{p} := \mathcal{P}(\boldsymbol{w}; \mathcal{C}(\boldsymbol{\beta}))$, and for brevity let $\boldsymbol{s} := (1-\alpha)\boldsymbol{w} + \alpha\boldsymbol{v}$. We then define the following sets,

$$\mathcal{P}_{inc} := \{i \in [n] : p_i > w_i\}\,, \quad \mathcal{P}_{dec} := \{i \in [n] : p_i \leq w_i\}\,,$$
$$\mathcal{S}_{inc} := \{i \in [n] : s_i > w_i\}\,, \quad \mathcal{S}_{dec} := \{i \in [n] : s_i \leq w_i\}\,.$$

The subscripts *inc* and *dec* correspond to the relative change in the weights before and after the corresponding update - whether they *increase* or *decrease*, respectively.

We first require the following corollary, which follows naturally from Theorem 16.

Corollary 21. *If $i \in \mathcal{P}_{inc}$ then $p_i = \beta_i$.*

Proof. Recall that Theorem 16 states that $\boldsymbol{p}$ is such that for $i = 1, \ldots, n$,

$$p_i = \max\{\beta_i; \lambda w_i\}\,,$$

where $\lambda = \frac{1-\sum_{j\in\Psi}\beta_j}{1-\sum_{j\in\Psi} w_j}$ is a normalizing constant. We first establish that $\lambda \leq 1$. Suppose $\lambda > 1$, then this implies $\sum_{i\in\Psi} w_i > \sum_{i\in\Psi}\beta_i$. In this case there must exist $i \in \Psi$ such that $w_i > \beta_i$. However if $\lambda > 1$ then $\lambda w_i > w_i > \beta_i$, but since $i \in \Psi$ then $p_i = \beta_i$, which must be greater than λw_i by Theorem 16. This leads to a contradiction and thus our supposition that $\lambda > 1$ is false.

The form of $\boldsymbol{p}$ implies that $i \in \mathcal{P}_{inc}$ if and only if $w_i < \beta_i$, since if $w_i \geq \beta_i$ then this implies that either $p_i = \beta_i \leq w_i$ or $p_i = \lambda w_i \leq w_i$, and in both of

these cases i must be in $\mathcal{P}_{dec}$. It then follows that if $i \in \mathcal{P}_{inc}$ then $p_i = \beta_i$ since otherwise $p_i = \lambda w_i \leq w_i < \beta_i$ which is a contradiction. □

Recall that it is assumed that $\boldsymbol{w} \neq \boldsymbol{v}$ and thus the definition of $\boldsymbol{s}$ implies that $\mathcal{S}_{inc}$ is non-empty. We use this fact in the following two lemmas. The first states that if a weight w_i were to increase after the projection update, then it would always increase after the weight-sharing update.

Lemma 22. *$\mathcal{P}_{inc} \subseteq \mathcal{S}_{inc}$.*

Proof. For any $i \in [n]$ we have

$$s_i - w_i = (1-\alpha)\, w_i + \alpha v_i - w_i = \alpha\,(v_i - w_i)\ ,$$

and it follows that $i \in \mathcal{S}_{inc}$ if and only if $w_i < v_i$. Using Corollary 21 we conclude that if $i \in \mathcal{P}_{inc}$, then $w_i < p_i = \beta_i = \alpha v_i < v_i$ and then i must also be in $\mathcal{S}_{inc}$. □

Lemma 23. *$\|\boldsymbol{p} - \boldsymbol{w}\|_1 = 2\sum_{i\in\mathcal{P}_{inc}}(p_i - w_i)$, and $\|\boldsymbol{s} - \boldsymbol{w}\|_1 = 2\sum_{i\in\mathcal{S}_{inc}}(s_i - w_i)$.*

Proof. We prove the first equality by observing that

$$\|\boldsymbol{p} - \boldsymbol{w}\|_1 = \sum_{i=1}^{n} |p_i - w_i| = \sum_{i\in\mathcal{P}_{inc}} (p_i - w_i) + \sum_{i\in\mathcal{P}_{dec}} (w_i - p_i)\,,$$

and since the total weight does not change after an update (i.e., $\sum_{i=1}^n p_i = \sum_{i=1}^n w_i$), necessarily we have $\sum_{i\in\mathcal{P}_{inc}}(p_i - w_i) = \sum_{i\in\mathcal{P}_{dec}}(w_i - p_i)$. Since $\sum_{i=1}^n s_i = \sum_{i=1}^n w_i$, the same argument can be used to prove the second claim. □

Proof of Theorem 20. Using Corollary 21, and the definition of $\boldsymbol{s}$, we have for $i \in \mathcal{P}_{inc}$,

$$s_i - w_i = (1-\alpha)w_i + \alpha v_i - w_i = \alpha(v_i - w_i) = \beta_i - \alpha w_i = p_i - \alpha w_i > p_i - w_i\,, \quad (4.30)$$

where the inequality arises from the fact that $\alpha < 1$. Finally combining this

inequality with Lemmas 22 and 23 gives

$$\begin{aligned}
\|\boldsymbol{p}-\boldsymbol{w}\|_1 &= 2\sum_{i\in\mathcal{P}_{inc}}(p_i-w_i) && \text{(Lemma 23)}\\
&< 2\sum_{i\in\mathcal{P}_{inc}}(s_i-w_i) && \text{(Equation 4.30)}\\
&\leq 2\sum_{i\in\mathcal{S}_{inc}}(s_i-w_i) && \text{(Lemma 22)}\\
&= \|\boldsymbol{s}-\boldsymbol{w}\|_1 \,. && \text{(Lemma 23)}
\end{aligned}$$

□

Thus if one has to pay to update weights, projection is the economical choice. It is illustrative to demonstrate how large the gap between the "cost" incurred by the projection and sharing updates can be. We will do so for the Fixed-Share update, that is, we set $\boldsymbol{v}=\frac{1}{n}\mathbf{1}$ and $\mathcal{C}(\boldsymbol{\beta})=[\frac{\alpha}{n},1]\cap\Delta_n$. Consider the following example, let

$$w_i=\begin{cases}\frac{\alpha}{n-1}, & i=1,\ldots,n-1\\ 1-\alpha, & i=n\,.\end{cases}$$

Observe that since all constraints of $\mathcal{C}(\boldsymbol{\beta})$ are satisfied, $\mathcal{P}(\boldsymbol{w};\mathcal{C}(\boldsymbol{\beta}))=\boldsymbol{w}$ and thus $\|\mathcal{P}(\boldsymbol{w};\mathcal{C}(\boldsymbol{\beta}))-\boldsymbol{w}\|_1=0$ and we incur no costs for projecting in this

instance. Meanwhile observe that $\mathcal{S}_{inc} = \{1, \ldots, n-1\}$ and thus

$$\begin{aligned}
\|\boldsymbol{s} - \boldsymbol{w}\|_1 &= 2 \sum_{i \in \mathcal{S}_{inc}} s_i - w_i \\
&= 2 \sum_{i=1}^{n-1} (1-\alpha)\, w_i + \frac{\alpha}{n} - w_i \\
&= \frac{2\alpha}{n} \sum_{i=1}^{n-1} (1 - n w_i) \\
&= \frac{2\alpha}{n} \sum_{i=1}^{n-1} \left(1 - \frac{\alpha n}{n-1}\right) \\
&= \frac{2\alpha(n-1)}{n} - 2\alpha^2 \,, \qquad (4.31)
\end{aligned}$$

which is maximized for $\alpha = \frac{n-1}{2n}$, the limit of which as $n \to \infty$ is $\frac{1}{2}$. In practice of course α is often tuned to be a lot smaller than $\frac{1}{2}$ in which case (4.31) is lower-bounded by α itself for all $\alpha \leq \frac{n-1}{2n}$. Nevertheless it is perhaps surprising that the cost of sharing can approach $\frac{1}{2}$ while the cost of projecting remains zero, especially given that the ℓ^1-diameter of the simplex is 2.

4.5 A Geometrically-Decaying Mixing Scheme for MPP

In this section we look more closely at Share-θ. We show that it is in fact a new type of *decaying* MPP mixing scheme which corresponds to the circadian specialist algorithm with Markov prior.

Recall that the previous best known mixing scheme for MPP is the decaying scheme (4.6). Observe that in (4.6) the decay (with the "distance" to the current trial t) follows a power-law, and that computing (4.6) exactly takes $\mathcal{O}(nt)$ time per trial. We now derive an explicit MPP mixing scheme from the updates (4.15) and (4.16) of Share-θ. Observe that if we define $\dot{\boldsymbol{w}}^0 := \frac{\mathbf{1}}{n}$, then an iterative expansion of (4.16) on any trial t gives $\boldsymbol{v}^t = \sum_{q=0}^{t-1} \theta^{[q \neq 0]} (1-\theta)^{t-q-1} \dot{\boldsymbol{w}}^q$, from

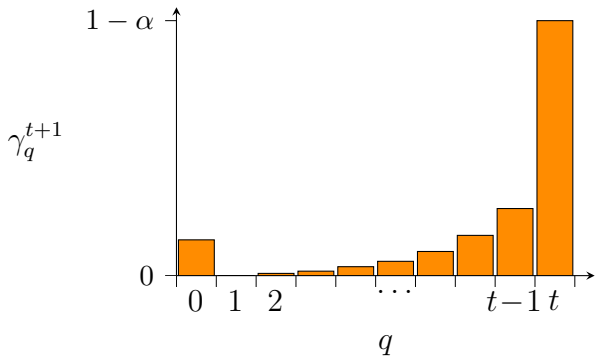


Figure 4.4: An illustration of the geometrically-decaying mixing scheme (4.32) which corresponds to the circadian specialists algorithm with Markov prior.

which (4.15) implies $\boldsymbol{w}^{t+1} = (1-\alpha)\dot{\boldsymbol{w}}^t + \alpha \boldsymbol{v}^t = \sum_{q=0}^{t} \gamma_q^{t+1} \dot{\boldsymbol{w}}^q$, where

$$\gamma_q^{t+1} = \begin{cases} 1-\alpha & q = t \\ \theta(1-\theta)^{t-q-1}\alpha & 1 \le q < t \\ (1-\theta)^{t-1}\alpha & q = 0\,. \end{cases} \cdot \tag{4.32}$$

Note that (4.32) is a valid mixing scheme since for all t, $\sum_{q=0}^{t} \gamma_q^{t+1} = 1$. The Share-$\theta$ update is therefore a new kind of decaying mixing scheme. In this new scheme the decay is *geometric*, and can therefore be computed efficiently, requiring only $\mathcal{O}(n)$ time and space per trial as we have shown. Furthermore we have shown that with this scheme MPP has the improved regret bound (4.17).

Another interesting difference between the decaying schemes (4.32) and (4.6) is that when θ is small then (4.32) keeps γ_0^{t+1} relatively large initially and slowly decays this value as t increases. Intuitively by heavily weighting the initial uniform vector $\dot{\boldsymbol{w}}^0$ on each trial early on, the algorithm can "pick up" the weights of new experts easily (see Figure 4.4 for an illustration). Finally as in the case of PoDS-θ, if $m = k+1$, then with the optimal tuning of $\theta = 0$, this update reduces to the Fixed-Share update (4.2).

4.5.1 Revisiting Circadian Specialists

We now turn our attention to the previous best known result for tracking experts with memory (the circadian specialists algorithm with a Markov prior [2]).

For sleep/wake patterns $(\chi_1 \dots \chi_T)$ the Markov prior is a Markov chain on states $\{w, s\}$, defined by the initial distribution $\boldsymbol{\pi} = (\pi_w, \pi_s)$ and transition probabilities $P_{ij} := P(\chi_{t+1} = j | \chi_t = i)$ for $i, j \in \{w, s\}$. The algorithm with these inputs efficiently collapses one weight per specialist down to two weights per expert. These two weight vectors, which we denote $\boldsymbol{a}_t$ and $\boldsymbol{s}_t$, represent the total weight of all awake and sleeping specialists associated with each expert, respectively. Note that the vectors $\boldsymbol{a}_t$ and $\boldsymbol{s}_t$ are not in Δ_n, but rather the vector $(\boldsymbol{a}_t, \boldsymbol{s}_t) \in \Delta_{2n}$ and the "awake vector" $\boldsymbol{a}_t$ gets normalized upon prediction. The weights are initialized by setting $\boldsymbol{a}_1 = \pi_w \frac{\mathbf{1}}{n}$, and $\boldsymbol{s}_1 = \pi_s \frac{\mathbf{1}}{n}$. The update[4] of these weights after receiving the true label y^t is given by

$$a_i^{t+1} = P_{ww} \frac{a_i^t e^{-\eta \ell_i^t} (\sum_{j=1}^n a_j^t)}{\sum_{j=1}^n a_j^t e^{-\eta \ell_j^t}} + P_{sw} s_i^t \tag{4.33}$$

and

$$s_i^{t+1} = P_{ws} \frac{a_i^t e^{-\eta \ell_i^t} (\sum_{j=1}^n a_j^t)}{\sum_{j=1}^n a_j^t e^{-\eta \ell_j^t}} + P_{ss} s_i^t \tag{4.34}$$

for $i = 1, \dots, n$. Recall that the authors of [2] proved that an MPP mixing scheme implicitly induces a prior over circadian specialists. The following states that the Markov prior is induced by (4.32).

Proposition 24. *Let $0 < \alpha < 1$, and $0 < \theta < 1$. Then the circadian specialists algorithm with Markov prior parameterized with $P_{sw} = \theta$, $P_{ws} = \alpha$, $\pi_w = \frac{\theta}{\alpha+\theta}$, and $\pi_s = \frac{\alpha}{\alpha+\theta}$ is equivalent to Share-θ parameterized with α and θ.*

Proof. It suffices to show that

$$\frac{a_i^t}{\sum_{j=1}^n a_j^t} = w_i^t \,, \tag{4.35}$$

and

$$\frac{s_i^t}{\sum_{j=1}^n s_j^t} = v_i^t \tag{4.36}$$

for all t. Since the initial distribution, $\boldsymbol{\pi}$, of the Markov chain prior is taken to

[4] In [2] the algorithm is presented in terms of probabilities with the log loss. Here we give the update generalized to (c, η)-realizable losses.

be the stationary distribution, the detailed balance equation, $P_{ws}\pi_w = P_{sw}\pi_s$, holds for all trials.

It is therefore straightforward to show that $\sum_{i=1}^n a_i^t = \pi_w$ and $\sum_{i=1}^n s_i^t = \pi_s$ for all t. Letting $\alpha = P_{ws}$, and $\theta = P_{sw}$, we proceed to prove that (4.35) and (4.36) hold simultaneously for all t by induction. The case for $t = 1$ is trivial. Then by induction on t for $t \geq 1$,

$$\begin{aligned}
\frac{a_i^{t+1}}{\pi_w} &= P_{ww}\frac{a_i^t e^{-\eta\ell_i^t}}{\sum_{j=1}^n a_j^t e^{-\eta\ell_j^t}} + \frac{P_{sw}}{\pi_w}s_i^t \\
&= P_{ww}\frac{a_i^t e^{-\eta\ell_i^t}}{\sum_{j=1}^n a_j^t e^{-\eta\ell_j^t}} + \frac{P_{ws}}{\pi_s}s_i^t \\
&= P_{ww}\dot{w}_i^t + P_{ws}v_i^t && \text{(induction)} \\
&= (1-\alpha)\dot{w}_i^t + \alpha v_i^t \\
&= w_i^{t+1}\,,
\end{aligned}$$

and similarly

$$\begin{aligned}
\frac{s_i^{t+1}}{\pi_s} &= \frac{P_{ws}\pi_w}{\pi_s}\frac{a_i^t e^{-\eta\ell_i^t}}{\sum_{j=1}^n a_j^t e^{-\eta\ell_j^t}} + P_{ss}\frac{s_i^t}{\pi_s} \\
&= P_{sw}\frac{a_i^t e^{-\eta\ell_i^t}}{\sum_{j=1}^n a_j^t e^{-\eta\ell_j^t}} + P_{ss}\frac{s_i^t}{\pi_s} \\
&= P_{sw}\dot{w}_i^t + P_{ss}v_i^t && \text{(induction)} \\
&= \theta\dot{w}_i^t + (1-\theta)v_i^t \\
&= v_i^{t+1}\,.
\end{aligned}$$

We therefore conclude by the inductive argument that (4.35) and (4.36) hold for all $t \geq 1$. □

The proof amounts to showing for all t that $\frac{\boldsymbol{a}_t}{\pi_w} = \boldsymbol{w}^t$ and $\frac{\boldsymbol{s}_t}{\pi_s} = \boldsymbol{v}^t$. The Markov prior on circadian specialists therefore corresponds to a geometrically-decaying MPP mixing scheme! Note however that we have proved a slightly tighter regret bound for this algorithm in Theorem 15.

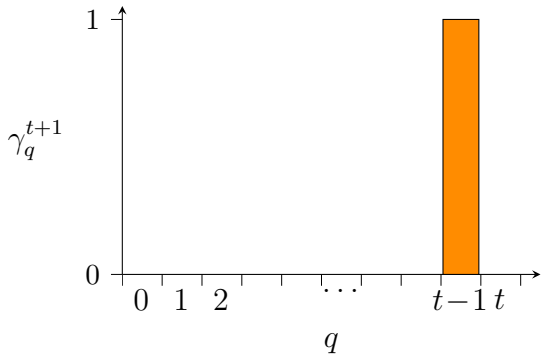


Figure 4.5: The "mixing scheme" of Share-θ introduced by setting $\alpha = \theta = 1$. Note that this corresponds to the setting where $m = 2$, and $k = T - 1$. That is, we switch back and forth on every trial between two good experts.

We have shown the association between Share-θ and the circadian specialists algorithm with Markov prior [2]. We showed that α corresponds to the parameter P_{ws} and θ corresponds to P_{sw}. We derived a different tuning from that of [2] however, with $\theta = \frac{k-m+1}{(m-1)(T-2)}$. There is an intuitive, yet perhaps surprising scenario worth considering with this tuning. Consider the case that both α and θ are set to 1. In this case our update becomes

$$\begin{aligned} \boldsymbol{w}^{t+1} &= (1-\alpha)\dot{\boldsymbol{w}}^t + \alpha \boldsymbol{v}^t \\ &= (1-\alpha)\dot{\boldsymbol{w}}^t + \alpha\left((1-\theta)\boldsymbol{v}^{t-1} + \theta\dot{\boldsymbol{w}}^{t-1}\right) \\ &= \dot{\boldsymbol{w}}^{t-1}\,. \end{aligned} \tag{4.37}$$

At first the idea of setting $\boldsymbol{w}^t = \dot{\boldsymbol{w}}^{t-2}$ seems strange, as the choice of weights on even-numbered trials ignores data from all previous odd-numbered trials and vice versa (see Figure 4.5). However, if we recall the optimal tuning of α and θ in terms of k, m, and T, then this update becomes intuitive. Indeed, since $\alpha = \frac{k}{T-1}$ then this implies that $k = T - 1$, that is, we switch on every trial. Additionally, $\theta = \frac{k-m+1}{(m-1)(T-2)}$ with $k = T - 1$ implies that $m = 2$. Therefore, the sequence for which setting $\alpha = \theta = 1$ is optimal is the sequence where only two experts perform well, and we switch back and forth between them on every trial. If we had prior information about this "two good experts" learning problem, then the update (4.37) and ignoring "odd" trials on "even" trials makes sense.

Chapter 5

Adaptive Long-Term Memory

5.1 Introduction to the Chapter

In this chapter we extend the model of non-stationary online learning with memory. In this model we study the problem of learning a switching sequence with localized periods of "memory." We will present our results in the adversarial contextual bandits setting, but our methods and results will be inspired by and directly applicable to the methods studied in Chapter 4 in the experts setting. We briefly introduced contextual bandits in Section 2.4 as a natural extension to the problem of prediction with expert advice. Recall that in this setting the `learner` is placed in an environment such that on each trial `nature` presents a context vector to the `learner` who has access to a number of *policies* which, given this context vector, output distributions over the available actions. In this case, the policies directly correspond to experts. Indeed, as in the prediction with expert advice framework, the `learner` selects an action based on the advice of these policies, before receiving feedback from `nature`. This feedback could be in the full-information setting, where the `learner` receives the losses of all actions on each trial, or the partial-information (bandit) setting, where the `learner` only receives the loss of its chosen action. We cover both cases in our presented algorithm and regret bound, but focus primarily on the partial-information setting in our discussion and in the experiments in Section 5.7.

We extend the model of switching with memory by presenting a two-layer long-term memory (LTM) strategy to efficiently remember previously good policies in both the short- and long-term. The set of policies may be very large and highly redundant, so we may expect that only a few policies will be relevant over the entire learning problem. In contrast with previous results on switching with memory, the novel switching model that we introduce is the assumption that the sequence of trials may be segmented into "epochs," where within each epoch is an even (possibly much) smaller set of relevant policies.

Consider, for example, an advertiser who manages a popular news website on which adverts will be displayed to users. Over time a sequence of users will visit the website, drawn primarily by the current trending news. The advertiser's available actions correspond to a set of possible adverts to be displayed to users. If the user clicks on the advert, then the loss incurred will be zero, and one otherwise. Of course, the advertiser wishes to display an advert that will be clicked on more often.

In order to aid the advertiser in their decision, they have access to a set of recommended selections from a prescribed set of policy functions. Each policy function makes its recommendation based on the received context vector, which could, for example, correspond to demographic information about the user, past behavior of the user, and so on. The policy functions themselves could be trained machine learning models on past data from previous advertising campaigns, and one may expect that some sets of policies are more suitable for some particular demographics of users.

As the topic of the *current* headline news may change relatively frequently over time (perhaps daily or weekly) then the type of user drawn to that website, and thus the relevant policy, may also change over time. Due to the changing nature of the news cycles, where updates or breaking news on a particular story may dominate temporarily, our assumption is not that there will be a unique "best" policy for the lifetime of the website's advertising campaign, but rather there will be a small set of "good" policies that will be relevant during

the campaign.

A further temporal element of this problem assumes that this small set of good policies will itself change over time. That is, we expect new stories of interest to emerge gradually, drawing different types of visitors corresponding to different policies. Indeed, on this longer time scale, as different news stories emerge, they may be of public interest for several days or weeks, interspersed with updates of other ongoing new stories, corresponding to different policies entering our pool of relevant policies (perhaps after many weeks/months of not being relevant) within which we are switching, before becoming irrelevant again.

The standard model of switching with memory corresponds to an assumption that previously relevant policies will once again be relevant in the future, forming a fixed, small pool of relevant policies. We studied this memory model in Chapter 4 in the context of prediction with expert advice. In this chapter, we refine the memory model such that this pool of relevant policies itself *evolves* over time. We thus have two layers of memory - in the outer layer there is the set of global relevant policies, and in the inner layer there is the "local" pool of relevant policies, which is a subset of this global pool. We call this model adaptive LTM.

5.1.1 Notation

We first introduce some notation specific to this chapter. The symmetric difference of sets A and B is denoted as $A \triangle B$. Given an n-dimensional vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and $m \leq n$ we define $\boldsymbol{x}^{|m}$ to be the prefix of $\boldsymbol{x}$ consisting of the first m elements $(x_1, x_2, \ldots, x_m)$. We denote vector-valued functions by bold letters. Given a vector-valued function $\boldsymbol{f}$ and some x in its domain, we denote the i^{th} component of $\boldsymbol{f}(x)$ as $f_i(x)$. For some integer $A \in \mathbb{N}$ we let $[A] := \{1, 2, \ldots, A\}$.

Protocol 5 The Contextual Protocol (Full and Partial Information)

for $t \leftarrow 1$ to T **do**
  `Nature` reveals context $\boldsymbol{x}^t \in \mathcal{X}$
  `Learner` observes policy advice $h(\boldsymbol{x}^t)$ for $h \in \mathcal{H}$
  `Learner` selects action $a^t \in [A]$

▷ `Full Information`
  `Nature` reveals losses $\boldsymbol{\ell}^t \in [0,1]^A$

▷ `Partial Information`
  `Nature` reveals loss $\ell^t_{a^t} \in [0,1]$

  `Learner` incurs $\ell^t = \ell^t_{a^t}$
end for

5.2 Background and Related Work

In this chapter we will consider the problem of adaptive LTM in both the full-information and the partial-information (bandit) setting.

Our model is described in Protocol 5. On any given trial $t \in [T]$ `nature` announces a context vector $\boldsymbol{x}^t \in \mathcal{X}$. The `learner` then chooses an action a^t from A possible actions based on advice generated by the set of policies $\mathcal{H}$. We let $N := |\mathcal{H}|$. A policy $h : \mathcal{X} \to \Delta_A$ is a mapping from a context vector to a distribution over actions. In the full-information setting the learner observes the loss vector $\boldsymbol{\ell}^t \in [0,1]^A$, and in the partial-information case the learner observes only the loss $\ell^t_{a^t} \in [0,1]$. In both cases the learner incurs $\ell^t_{a^t}$, the loss of its chosen action. Our discussion and experiments will primarily focus on the partial-information case.

In proving our regret bound in this setting, we will assume that `nature` is a deterministic oblivious adversary (see e.g., [84, Section 5.1]). That is, we assume that `nature` selects $\boldsymbol{x}^t$ and $\boldsymbol{\ell}^t$ for $t \in T$ before learning begins and that these are, of course, unknown to the `learner`. Given a comparison vector of policies $\boldsymbol{h} \in \mathcal{H}^T$ we define the regret with respect to that vector as

$$\mathcal{R}(\boldsymbol{h}) := \mathbb{E}\left[\sum_{t=1}^{T} \ell^t_{a^t}\right] - \mathop{\mathbb{E}}_{b^t \sim h_t(\boldsymbol{x}^t)}\left[\sum_{t=1}^{T} \ell^t_{b^t}\right], \tag{5.1}$$

where the first expectation is with respect to the possible randomization of the `learner`'s algorithm.

In this chapter we will prove bounds of the following form:

$$\mathcal{R}(\boldsymbol{h}) \leq \mathcal{O}\left(\sqrt{C(\boldsymbol{h})T}\right) \tag{5.2}$$

in the full-information setting and

$$\mathcal{R}(\boldsymbol{h}) \leq \mathcal{O}\left(\sqrt{AC(\boldsymbol{h})T}\right) \tag{5.3}$$

in the partial-information setting. The quantity $C(\boldsymbol{h})$ acts as an information-theoretic complexity measure of the comparison vector $\boldsymbol{h}$. Our discussion on regret bounds will therefore focus on the quantity $C(\boldsymbol{h})$.

The simplest bandit problem is the (non-contextual) multi-armed bandit (MAB) problem, where regret is measured with respect to a sequence of actions. This (non-switching) problem was pioneered, in the adversarial case, by the EXP3 algorithm of [85]. This paper also used the mechanics of Fixed Share [30] to develop a switching MAB algorithm: EXP3.S. A special case of the GABA algorithms of [86] was to incorporate memory into MABs. However, observe that since regret bounds in this setting are of the form $\mathcal{R}(\boldsymbol{h}) \leq \mathcal{O}(\sqrt{AC(\boldsymbol{h})T})$, the leading factor of A in the square root means that memory results for MABs are, in general, much less impactful than those for full-information.

The paper [57] managed to get impactful results for MABs by restricting itself to sparse problems. This paper also gave memory algorithms for the full-information case but with weaker regret bounds than [2]. In this work, we get impactful results for LTM in the partial-information setting by considering policy-based contextual bandits, pioneered by the EXP4 algorithm of [85]. In this setting, as we have seen, the regret is measured with respect to a sequence of *policies* rather than actions. Note that while we present our algorithm and regret bound in the contextual bandits setting, it is straightforward to derive the corresponding regret bounds in the prediction with expert advice setting

with for example (c, η)-realizable losses as we considered in Chapter 4.

In [85] the seminal contextual bandit algorithm EXP4 was given, which considered the case that the policy vector is constant over time, that is, $\boldsymbol{h} = (h, h, \ldots, h) \in \mathcal{H}^T$ for some policy $h \in \mathcal{H}$. In this setting the EXP4 algorithm obtains

$$C(\boldsymbol{h}) = \log N \,.$$

In the non-stationary setting, the policy vector has a richer structure. We first review the case of (non-memory) switching, and then introduce our two-layer memory model that allows the `learner` to adapt more quickly to a new policy performing well if it has been observed before, even a very long time ago.

We define $K(\boldsymbol{h}) := \sum_{t=1}^{T-1} [\![h_t \neq h_{t+1}]\!]$ to be the number of switches in the policy vector. It is straightforward to show that an adaptation of the algorithm EXP3.S of [85] for the contextual bandit setting gives an algorithm with

$$C(\boldsymbol{h}) = K(\boldsymbol{h}) \log\left(\frac{T}{K(\boldsymbol{h})}\right) + K(\boldsymbol{h}) \log N \,, \tag{5.4}$$

which is analogous to the Fixed Share bound discussed in previous chapters. Given this regret bound of $\mathcal{R}(\boldsymbol{h}) \leq \mathcal{O}(\sqrt{AC(\boldsymbol{h})T})$, observe that when the policies in this vector do not change too often, i.e., $C(\boldsymbol{h}) \leq o(T)$, the regret bound is non-vacuous.

Similarly, in the switching with memory setting, as studied in Chapter 4 in the experts setting, if we define $M(\boldsymbol{h}) := |\cup_{t=1}^{T} \{h_t\}|$ to be the size of the small pool, then bounds of the form $\mathcal{R}(\boldsymbol{h}) \leq \mathcal{O}(\sqrt{AC(\boldsymbol{h})T})$ are achievable with

$$C(\boldsymbol{h}) = M(\boldsymbol{h}) \log\left(\frac{N}{M(\boldsymbol{h})}\right) + K(\boldsymbol{h}) \log\left(\frac{M(\boldsymbol{h})T}{K(\boldsymbol{h})}\right) \tag{5.5}$$

with adaptations of the algorithms of [1, 2] and the methods studied in Chapter 4 to the contextual bandit setting.

Recall that for the problem of switching with memory, the salient feature of the regret bound is that we pay less per switch (compared to the non-memory

switching problem) by assuming that the comparator sequence is made of only a small pool of experts (policies). As discussed, in this work we go even further by allowing this pool of experts to change over time, hence "adaptive LTM".

An observed obstacle to the contextual bandit setting is that in some applications, the set of policies required may be enormous. In such cases, algorithms such as EXP4 and its extensions discussed above (whose computation time scales linearly with the number of policies) are not practical. An active area of research in contextual bandits is developing *oracle-efficient* algorithms, which assume access to an offline oracle for a specific associated optimization problem (see e.g., [87, 88, 89, 90]) to avoid computation time being linear in the number of policies. The first oracle-efficient result with sub-linear regret for the adversarial contextual bandits setting was [89], which also considered the switching (without memory) setting. While sub-linear, the regret bounds of oracle-efficient algorithms for adversarial contextual bandits are generally worse than that of, e.g., EXP4 and its extensions. In [90], however, an improved oracle-efficient method was given for the switching (without memory) setting with a bound that is very close to (5.4) for the switching variant of EXP4. As far as we know, the problem of switching with memory has yet to be addressed for oracle-efficient adversarial contextual bandit algorithms, let alone our new, significantly more complex adaptive LTM model.

For our algorithm, we will aim to have a per-trial time complexity of $\mathcal{O}(N)$. Conceptually, given our adaptive LTM model's complexity (across time), a time complexity that does not grow with time is a surprising result. Therefore, we will leave the question of whether our methods can be applied in an oracle-efficient manner as an open question. We also point out that our results improve over the state of the art in the full-information setting and that in the experts setting our time-complexity matches that of the algorithms and methods studied in Chapter 4 for either switching without memory or switching with (non-adaptive) memory.

5.3 Adaptive Long-Term Memory

We now formally introduce our model of adaptive LTM. This model can be viewed as a two-layered memory model, stratified into a "local" working memory, and a "global" long-term memory. The global long-term memory corresponds to a pool of policies $\mathcal{P} \subseteq \mathcal{H}$. The sequence of trials $t = 1, \ldots, T$ can be partitioned into *epochs*, and for epoch i the "working memory" corresponds to a set of policies denoted $\mathcal{P}^i \subseteq \mathcal{P}$. We thus assume that this epoch's pool, $\mathcal{P}^i$, corresponds to a "localized" working memory problem, analogous to the setting studied in Chapter 4. The top layer of memory across epochs ties these working memories together. This is reflected in our regret bound. The concept of epochs is introduced only to characterize the trial sequence and in the analysis of our algorithm to derive our regret bound. Our regret bound will hold for any partitioning of the trial sequence into epochs, and the `learner` does not know when one epoch ends and another begins.

Note that the bounds considered in Section 5.2 all scale with $C(\boldsymbol{h})$. In our stratified model, however, our bound will additionally depend on the chosen partitioning, $\mathcal{E}$, of $\boldsymbol{h}$ into epochs. We therefore first introduce the notation (and give an illustration in Figure 5.1) required to derive the quantity $C(\boldsymbol{h}, \mathcal{E})$ on which our bound will depend.

We define a partitioning of T trials into epochs, $\mathcal{E}$, as follows. A partitioning $\mathcal{E}$ consists of E epochs defined as $\mathcal{E} := (e^1, \ldots, e^E) \subseteq [T]$ where $1 = e^1 < e^2 < \ldots < e^E \leq T$ and conventionally we have $e^{E+1} := T + 1$. Given such notation for a partitioning into epochs, we now define the derived quantities on which $C(\boldsymbol{h}, \mathcal{E})$ will depend.

Definition 25. *Given a policy vector $\boldsymbol{h} \in \mathcal{H}^T$ and a partition $\mathcal{E}$ into epochs the **derived** quantities are:*

$$\begin{aligned} \boldsymbol{derive}(\boldsymbol{h}, \mathcal{E}) :=& (\mathcal{P}, M, \mathcal{P}^1, \ldots, \mathcal{P}^E, M^1, \ldots, M^E, \Phi, \\ & T^1, \ldots, T^E, K^1, \ldots, K^E) \,. \end{aligned} \tag{5.6}$$

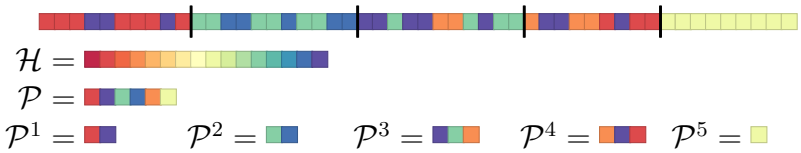


Figure 5.1: A toy example trial sequence of our adaptive LTM model. We show some of the derived quantities of $(\boldsymbol{h}, \mathcal{E})$. The policy vector $\boldsymbol{h} \in \mathcal{H}^{50}$ is divided into $E = 5$ epochs. The epoch structure $\mathcal{E}$ is shown by the black overlaid lines. Example derived quantities include: $T = 50$, $|\mathcal{H}| = N = 16$, $M = 6$, $M^2 = 2$, $M^3 = 3$, $\mathcal{E} = (1, 11, 22, 33, 42)$, $\Phi = 13$.

The quantities derived from $(\boldsymbol{h}, \mathcal{E})$ are defined at both the local epoch level and the global level. At the epoch level, we have the following. For epoch i we have:

- $\mathcal{P}^i := \bigcup_{t \in [e^i, e^{i+1})} \{h_t\}$ - the set of distinct policies in the epoch ("working memory").
- $M^i := |\mathcal{P}^i|$ - the cardinality of the epoch set.
- $K^i := 1 + \sum_{t \in [e^i, e^{i+1})} [\![h_t \neq h_{t+1}]\!]$ - the number of policy switches (+1) in the epoch.
- $T^i := e^{i+1} - e^i$ - the length of the epoch.

At the global level, we have the global pool $\mathcal{P} := \cup_{i \in [E]} \mathcal{P}^i$, its cardinality $M := |\mathcal{P}|$, the total number of trials $T = \sum_{i \in [E]} T^i$, and finally $\Phi := \sum_{i \in [E-1]} |\mathcal{P}^i \triangle \mathcal{P}^{i+1}|$, the number of new policies added or removed across epochs. See Figure 5.1 for an illustration and example of these quantities.

5.3.1 The AdaptLTM Regret Bound

Before we develop our algorithm for the problem of Adaptive LTM, which we will call AdaptLTM, we first give its regret bound in the following theorem. The bound is given for both the full-information and partial-information cases.

Theorem 26. *Algorithm AdaptLTM with policy set $\mathcal{H} \subseteq \Delta_A^{\mathcal{X}}$, $N := |\mathcal{H}|$ has for any policy vector $\boldsymbol{h} \in \mathcal{H}^T$ and any partition $\mathcal{E}$ into epochs with the derived*

quantities $(\mathcal{P}, \ldots, K^E) = \textbf{\textit{derive}}(\boldsymbol{h}, \mathcal{E})$ *(per Definition 25), with*

$$C = M \ln\left(\frac{N}{M}\right) + \Phi \ln\left(\frac{MT}{\Phi}\right) + \sum_{i \in [E]} K^i \ln\left(\frac{M^i T^i}{K^i}\right) \tag{5.7}$$

the following regret bounds:

$$\mathcal{R}(\boldsymbol{h}) \leq \mathcal{O}(\sqrt{ACT})$$

in the partial-information case and

$$\mathcal{R}(\boldsymbol{h}) \leq \mathcal{O}(\sqrt{CT})$$

in the full-information case, which are obtained by setting $\eta = \sqrt{\frac{2C}{AT}}$ *and* $\eta = \sqrt{\frac{2C}{T}}$ *respectively.*

The proof of Theorem 26 is deferred to Appendix B, as it depends on many properties developed in the remaining sections of this chapter.

Note that we can decompose C as $C = M \ln(N/M) + C^\dagger$, where

$$C^\dagger := \Phi \ln\left(\frac{MT}{\Phi}\right) + \sum_{i \in [E]} K^i \ln\left(\frac{M^i T^i}{K^i}\right).$$

Informally $M \ln(N/M)$ can be seen as the cost of learning the global pool $\mathcal{P}$, whilst $C^\dagger$ can be seen as the cost of learning the policy sequence $\{h_t \mid t \in [T]\}$ given the pool $\mathcal{P}$.

We will compare our bound to (5.5), which is the bound achieved for existing algorithms for switching with (non-adaptive) memory. Let this value be denoted $\hat{C} = M \ln(N/M) + K \ln(MT/K)$. Crucially then, we must compare $C^\dagger$ to $K \ln(MT/K)$. Observe that when the trial sequence has a single epoch then these terms are equal, and our bound reduces to the single epoch case (up to constant factors).

We now show how our bound improves over $\hat{C}$ when we have multiple epochs. In $\hat{C}$ each switch costs us the sum of two terms. Firstly $\ln(M)$,

which is informally the cost of learning the new policy from $\mathcal{P}$, and secondly $\ln(T/K)$, which is informally the cost of learning when the switch occurs. We observe that this second quantity is the logarithm of the inverse of the *average* switching rate. $C^\dagger$ improves both of these terms. Firstly, we pay an overhead of $\Phi \ln(MT/\Phi)$, then for each switch in epoch i we pay $\ln(M^i)$ instead of $\ln(M)$. The improvement here is clear since typically $M^i \ll M$. Secondly, we pay $\ln(T^i/K^i)$ instead of $\ln(T/K)$. This term, of course, corresponds to the logarithm of the inverse of the average switching rate *of that epoch*. Thus in a sense we adapt to the possibly different switching rates of each epoch. The improvement is quite interesting since for epochs with higher switching rates (than the global average) we pay a lot less (relatively) than paying the global average per switch, and for epochs with lower switching rates (relative to the global average) we do not pay much more. The stronger improvement, however, is in paying $\ln(M^i)$ rather than $\ln(M)$.

5.4 Policy Specialists

In the following sections we will develop our algorithm for the problem of adaptive LTM. As in previous works on switching with memory [2, 12], our algorithm is an efficient implementation of a circadian specialist algorithm. Indeed our method is an extension of the circadian specialist algorithm of [2], which we studied in Chapter 4. Recall that in our setting the specialist predictions are probability vectors (policies) over the set of actions, requiring a slightly more generalized specialist algorithm than previous works. This generalized specialist algorithm is given in Algorithm 6, which we call Policy Specialists. This algorithm will form the basis of our main algorithm, which we will call AdaptLTM.

Recall that the defining characteristic of these specialist algorithms is the notion of a circadian pattern of awake/asleep states, which we denote by a vector $\boldsymbol{\gamma} \in \{0,1\}^T$. In this case we have $\gamma_t = 1$ corresponding to a specialist being awake on trial t and $\gamma_t = 0$ corresponding to a specialist being asleep

Algorithm 6 POLICY SPECIALISTS

Input: $\eta > 0$; $\boldsymbol{w}^1 \in \Delta_{|\mathcal{S}|}$

for $t = 1, \ldots, \mathrm{T}$ **do**
  receive $\boldsymbol{x}^t \in \mathcal{X}$
  for $h \in \mathcal{H}$ **do**
    $\epsilon_h^t \leftarrow \sum_{\boldsymbol{\gamma} \in \{0,1\}^T} [\![\gamma_t = 1]\!] w_{(h,\boldsymbol{\gamma})}^t$
  end for
  $z^t \leftarrow \sum_{h \in \mathcal{H}} \epsilon_h^t$
  for $a \in [A]$ **do**
    $p_a^t \leftarrow \sum_{h \in \mathcal{H}} \frac{\epsilon_h^t h(\boldsymbol{x}^t)_a}{z^t}$
  end for
  select $a^t \sim \boldsymbol{p}^t$

▷ `Full Information`

  receive $\boldsymbol{\ell}^t \in [0,1]^A$
  for $h \in \mathcal{H}$ **do**
    $\hat{\ell}_h^t \leftarrow \sum_{a \in [A]} \ell_a^t h(\boldsymbol{x}^t)_a$
  end for

▷ `Partial Information`

  receive $\ell_{a^t}^t \in [0,1]$
  for $h \in \mathcal{H}$ **do**
    $\hat{\ell}_h^t \leftarrow \frac{\ell_{a^t}^t h(\boldsymbol{x}^t)_{a^t}}{p_{a^t}^t}$
  end for

  $\hat{z}^t \leftarrow \sum_{h \in \mathcal{H}} \epsilon_h^t e^{-\eta \hat{\ell}_h^t}$
  for $h \in \mathcal{H}$ **do**
    $\psi_h^t \leftarrow \frac{z^t e^{-\eta \hat{\ell}_h^t}}{\hat{z}^t}$
  end for
  for $(h, \boldsymbol{\gamma}) \in \mathcal{E}$ **do**
    $$w_{(h,\boldsymbol{\gamma})}^{t+1} \leftarrow \begin{cases} w_{(h,\boldsymbol{\gamma})}^t & \gamma_t = 0 \\ \psi_h^t w_{(h,\boldsymbol{\gamma})}^t & \gamma_t = 1 \end{cases}$$
  end for
end for

and abstaining from predicting. Our specialist set, which in this chapter we denote $\mathcal{S}$, will be the set of all policy-circadian pairs thus $\mathcal{S} = \mathcal{H} \times \{0,1\}^T$. A non-abstaining specialist predicts in accordance with its prescribed policy, h. We will index a circadian specialist by the tuple $(h, \boldsymbol{\gamma})$. Although we describe our algorithm as if we had prior knowledge of the number of trials, T, we will see that the algorithm does not need this knowledge a-priori.

POLICY SPECIALISTS takes as input a learning rate, η, and prior weight

$\boldsymbol{w}^1 \in \Delta_{|\mathcal{S}|}$. On each trial, $t = 1, \ldots, T$, the weighted average of the predictions of the non-abstaining specialists is computed (lines 3-9), with the sum of the awake weights taken on line 4. An action, a^t, is then drawn from the resulting probability distribution (line 10). Upon receiving information on the loss of that trial, the weights of the non-abstaining specialists are updated, re-normalizing such that our total weight is unchanged (lines 15-21). In the partial-information case we employ the standard trick of using inverse-propensity scoring to weight the loss of the chosen action (line 13) such that the quantity $\hat{\ell}_h^t$ is an unbiased estimator, i.e.,

$$\mathbb{E}[\hat{\ell}_h^t] = \sum_{a \in [A]} \mathbb{P}(a = a^t) \frac{\ell_a^t h(\boldsymbol{x}^t)_a}{p_a^t} = \sum_{a \in [A]} \ell_a^t h(\boldsymbol{x}^t)_a \,.$$

This method, introduced in [85] in the EXP4 algorithm, essentially allows for a reduction to the (full-information) Hedge setting [91], from the "exponential-weight" family of algorithms. For each policy $h \in \mathcal{H}$, we define the quantity ψ_h^t which combines the loss term and normalization factor, greatly simplifying our presentation and analysis. Observe that only the weights of awake specialists are updated (line 20).

As discussed, the set of specialists used in this algorithm is the set of circadian-policy (expert) pairs, which was the method introduced in [2] for the circadian specialists algorithm with simple Markov prior, which we studied in Chapter 4. Our algorithm, however, differs from that of [2] in two distinct ways. Firstly, as discussed, POLICY SPECIALISTS is a generalization in that each specialist predicts with a distribution. That is, on each trial the specialist $(h, \boldsymbol{\gamma})$ predicts $h(\boldsymbol{x}^t) \in \Delta_A$ if $\gamma_t = 1$ and abstains otherwise ($h(\boldsymbol{x}^t) = \square$). Secondly, and more importantly, we will show that our algorithm is more suited to the problem of adaptive LTM due to having a different "prior" weight over this set of specialists.

In the next section we will derive the generalized regret bound for POLICY SPECIALISTS for any choice of prior weight $\boldsymbol{w}^1$. In the following sections we

will then show how $\boldsymbol{w}^1$ can be chosen such that POLICY SPECIALISTS has a good regret bound (given in Theorem 26) for the problem of adaptive LTM, and can be implemented efficiently.

5.4.1 The POLICY SPECIALISTS Regret Bound

To prove a regret bound for POLICY SPECIALISTS it will be useful to describe the policy specialists of interest relative to the policy sequence $\boldsymbol{h} \in \mathcal{H}^T$ chosen by `nature`. That is, given $\boldsymbol{h}$, for every policy $h \in \mathcal{H}$ we define the circadian $\boldsymbol{c}^h \in \{0,1\}^T$ such that for all $t \in [T]$ we have

$$c_t^h := [\![h_t = h]\!] \,. \tag{5.8}$$

Note that $\boldsymbol{c}^h$ is a function of $\boldsymbol{h}$, and simply corresponds to the circadian pattern $\boldsymbol{\gamma} \in \{0,1\}^T$ of the particular specialist $(h, \boldsymbol{\gamma})$ required to predict in accordance with policy h on trials where $h_t = h$. This can be seen visually in Figure 4.1 from the previous chapter, as well as in Figure 5.3.

Given a policy sequence $\boldsymbol{h} \in \mathcal{H}^T$ we define the distribution $\boldsymbol{w}^* \in \Delta_{|\mathcal{S}|}$ to be such that:

$$w^*_{(h,\boldsymbol{\gamma})} := \begin{cases} \frac{1}{M} & h \in \mathcal{P} \text{ and } \boldsymbol{\gamma} = \boldsymbol{c}^h \\ 0 & \text{otherwise} \,. \end{cases}$$

For all $t \in [T]$ the relative entropy between $\boldsymbol{w}^*$ and $\boldsymbol{w}^t$ can then be written as

$$\begin{aligned} D(\boldsymbol{w}^*, \boldsymbol{w}^t) &= \sum_{(h,\boldsymbol{\gamma}) \in \mathcal{S}} w^*_{(h,\boldsymbol{\gamma})} \ln \left(\frac{w^*_{(h,\boldsymbol{\gamma})}}{w^t_{(h,\boldsymbol{\gamma})}} \right) \\ &= \sum_{h \in \mathcal{P}} \frac{1}{M} \ln \left(\frac{1}{M w^t_{(h,\boldsymbol{c}^h)}} \right) . \end{aligned}$$

We denote $MD(\boldsymbol{w}^*, \boldsymbol{w}^1)$ by $\bar{C}$. That is,

$$\bar{C} := \sum_{h \in \mathcal{P}} \ln \left(\frac{1}{M w^1_{(h,\boldsymbol{c}^h)}} \right) . \tag{5.9}$$

Note that on each trial $t \in [T]$ there is only one specialist from the set of

specialists $\{(h, \boldsymbol{c}^h) \mid h \in \mathcal{P}\}$ that is awake. Thus,

$$\begin{aligned}\frac{w^{t+1}_{(h,\boldsymbol{c}^h)}}{w^t_{(h,\boldsymbol{c}^h)}} &= [\![c^h_t = 0]\!] + [\![c^h_t = 1]\!]\psi^t_h \\ &= [\![h_t \neq h]\!] + [\![h_t = h]\!]\psi^t_h \,,\end{aligned}$$

so

$$\begin{aligned}D(\boldsymbol{w}^*, \boldsymbol{w}^t) - D(\boldsymbol{w}^*, \boldsymbol{w}^{t+1}) &= \sum_{h\in\mathcal{P}} \frac{1}{M} \ln\left(\frac{w^{t+1}_{(h,\boldsymbol{c}^h)}}{w^t_{(h,\boldsymbol{c}^h)}}\right) \\ &= \frac{1}{M}\ln\left(\psi^t_{h_t}\right),\end{aligned}$$

and hence due to the non-negativity of the relative entropy,

$$\begin{aligned}\bar{C} &= MD(\boldsymbol{w}^*, \boldsymbol{w}^1) \\ &\geq M\left(D(\boldsymbol{w}^*, \boldsymbol{w}^1) - D(\boldsymbol{w}^*, \boldsymbol{w}^{T+1})\right) \\ &= M\left(\sum_{t=1}^{T} D(\boldsymbol{w}^*, \boldsymbol{w}^t) - D(\boldsymbol{w}^*, \boldsymbol{w}^{t+1})\right) \\ &= \sum_{t=1}^{T} \ln\left(\psi^t_{h_t}\right).\end{aligned}$$

Since $\bar{C}$ is a fixed quantity and this holds for any sequence of losses, it holds in the expected case. Thus we have

$$\sum_{t=1}^{T} \mathbb{E}[\ln\left(\psi^t_{h_t}\right)] \leq \bar{C}\,. \tag{5.10}$$

To prove a bound for POLICY SPECIALISTS we must bound $\mathbb{E}[\ln\left(\psi^t_{h_t}\right)]$ for all $t \in [T]$ in both the full-information and partial-information case. Note first that by definition

$$\mathbb{E}\left[\ln\left(\psi^t_{h_t}\right)\right] = -\eta\mathbb{E}\left[\hat{\ell}^t_{h_t}\right] + \mathbb{E}\left[\ln\left(\frac{z^t}{\hat{z}^t}\right)\right], \tag{5.11}$$

so we shall now bound both terms on the right-hand side. First we have, in

the partial-information case, that for all $h \in \mathcal{H}$

$$\mathbb{E}[\hat{\ell}_h^t] = \sum_{a \in [A]} \mathbb{P}(a = a^t) \frac{\ell_a^t h(\boldsymbol{x}^t)_a}{p_a^t} = \sum_{a \in [A]} \ell_a^t h(\boldsymbol{x}^t)_a \tag{5.12}$$

so indeed, in both cases we have

$$\mathbb{E}[\hat{\ell}_{h_t}^t] = h_t(\boldsymbol{x}^t) \cdot \boldsymbol{\ell}^t \,. \tag{5.13}$$

We now turn to bounding $\mathbb{E}[\ln(z^t/\hat{z}^t)]$. Observe that we have

$$\frac{\hat{z}^t}{z^t} = \sum_{h \in \mathcal{H}} \frac{\epsilon_h^t}{z^t} e^{-\eta \hat{\ell}_h^t} \,.$$

Using the inequality $e^{-u} \leq 1 - u + u^2/2$ for $u \geq 0$ we then have

$$\begin{aligned} \frac{\hat{z}^t}{z^t} &\leq \sum_{h \in \mathcal{H}} \frac{\epsilon_h^t}{z^t} \left(1 - \eta \hat{\ell}_h^t + \frac{\eta^2 (\hat{\ell}_h^t)^2}{2}\right) \\ &= 1 - \eta \sum_{h \in \mathcal{H}} \frac{\epsilon_h^t}{z^t} \hat{\ell}_h^t + \frac{\eta^2}{2} \sum_{h \in \mathcal{H}} \frac{\epsilon_h^t}{z^t} (\hat{\ell}_h^t)^2 \,. \end{aligned}$$

Taking logarithms, noting that $\ln(1+u) \leq u$ for all $u \in \mathbb{R}$, and then taking expectations gives us

$$\mathbb{E}\left[\ln\left(\frac{z^t}{\hat{z}^t}\right)\right] \geq \eta \sum_{h \in \mathcal{H}} \frac{\epsilon_h^t}{z^t} \mathbb{E}\left[\hat{\ell}_h^t\right] - \frac{\eta^2}{2} \sum_{h \in \mathcal{H}} \frac{\epsilon_h^t}{z^t} \mathbb{E}\left[(\hat{\ell}_h^t)^2\right] \,. \tag{5.14}$$

We now bound the two sums on the right-hand side of (5.14). Note first that from (5.12) we have, in both the full-information and the partial-information case, that

$$\begin{aligned} \sum_{h \in \mathcal{H}} \frac{\epsilon_h^t}{z^t} \mathbb{E}\left[\hat{\ell}_h^t\right] &= \sum_{h \in \mathcal{H}} \frac{\epsilon_h^t}{z^t} \sum_{a \in [A]} \ell_a^t h(\boldsymbol{x}^t)_a \\ &= \sum_{a \in [A]} \ell_a^t p_a^t \\ &= \boldsymbol{p}^t \cdot \boldsymbol{\ell}^t \,. \end{aligned} \tag{5.15}$$

For the second sum, let us first define $A^\dagger := 1$ in the full-information case and $A^\dagger := A$ in the partial information case. First, note that in the full-information case since $\hat{\ell}_h^t \in [0,1]$ we then have

$$\sum_{h\in\mathcal{H}} \frac{\epsilon_h^t}{z^t} \mathbb{E}\left[(\hat{\ell}_h^t)^2\right] \le \sum_{h\in\mathcal{H}} \frac{\epsilon_h^t}{z^t} = 1 = A^\dagger\,, \tag{5.16}$$

and similarly in the partial-information case since $\ell_a^t \in [0,1]$ and $h(\boldsymbol{x}^t)_a \in [0,1]$ we have

$$\begin{aligned}
\mathbb{E}\left[(\hat{\ell}_h^t)^2\right] &= \sum_{a\in[A]} \mathbb{P}(a = a^t) \left(\frac{\ell_a^t h(\boldsymbol{x}^t)_a}{p_a^t}\right)^2 \\
&= \sum_{a\in[A]} \frac{(\ell_a^t h(\boldsymbol{x}^t)_a)^2}{p_a^t} \\
&\le \sum_{a\in[A]} \frac{h(\boldsymbol{x}^t)_a}{p_a^t}
\end{aligned}$$

and thus

$$\begin{aligned}
\sum_{h\in\mathcal{H}} \frac{\epsilon_h^t}{z^t} \mathbb{E}\left[(\hat{\ell}_h^t)^2\right] &\le \sum_{a\in[A]} \frac{1}{p_a^t} \sum_{h\in\mathcal{H}} \frac{\epsilon_h^t}{z^t} h(\boldsymbol{x}^t)_a \\
&= \sum_{a\in[A]} \frac{p_a^t}{p_a^t} \\
&= A \\
&= A^\dagger\,.
\end{aligned} \tag{5.17}$$

Substituting (5.15), (5.16), and (5.17) into (5.14) and then substituting that and (5.13) into (5.11) gives

$$\mathbb{E}[\ln(\psi_{h_t}^t)] \ge \eta\left(\boldsymbol{p}^t \cdot \boldsymbol{\ell}^t - h_t(\boldsymbol{x}^t) \cdot \boldsymbol{\ell}^t\right) - \frac{\eta^2 A^\dagger}{2}\,.$$

Summing over all trials and rearranging then gives

$$\begin{aligned}\mathcal{R}(\boldsymbol{h}) &= \sum_{t=1}^{T} \left(\boldsymbol{p}^t \cdot \boldsymbol{\ell}^t - h_t(\boldsymbol{x}^t) \cdot \boldsymbol{\ell}^t\right) \\ &\leq \frac{1}{\eta} \sum_{t=1}^{T} \mathbb{E}[\ln\left(\psi_{h_t}^t\right)] + \frac{\eta A^\dagger T}{2}\,,\end{aligned}$$

which combined with (5.10) gives

$$\mathcal{R}(\boldsymbol{h}) \leq \frac{\bar{C}}{\eta} + \frac{\eta A^\dagger T}{2}\,.$$

Thus the regret bound depends entirely on $\bar{C} = MD(\boldsymbol{w}^*, \boldsymbol{w}^1)$, and therefore our choice of $\boldsymbol{w}^1$. With an optimal tuning of $\eta = \sqrt{2\bar{C}/A^\dagger T}$ this gives a regret bound of $\mathcal{R}(\boldsymbol{h}) \leq \mathcal{O}(\sqrt{A^\dagger \bar{C} T})$.

5.4.2 Initial Weighting

We have shown that the regret bound depends entirely on our choice of $\boldsymbol{w}^1 \in \Delta_{|\mathcal{S}|}$. We will specify $\boldsymbol{w}^1$ by defining a function $f : \{0,1\}^T \to [0,1]$ such that $\sum_{\boldsymbol{\gamma}\in\{0,1\}^T} f(\boldsymbol{\gamma}) = 1$. Intuitively this function is just a prior distribution on circadian patterns.

Given this choice of $f(\boldsymbol{\gamma})$ we then, for all policy/circadian pairs $(h, \boldsymbol{\gamma}) \in \mathcal{S}$ define

$$w^1_{(h,\boldsymbol{\gamma})} := \frac{1}{N} f(\boldsymbol{\gamma})\,. \tag{5.18}$$

Note that by definition, we have

$$\sum_{(h,\boldsymbol{\gamma})\in\mathcal{S}} w^1_{(h,\boldsymbol{\gamma})} = \sum_{h\in\mathcal{H}} \frac{1}{N} \sum_{\boldsymbol{\gamma}\in\{0,1\}^T} f(\boldsymbol{\gamma}) = \sum_{h\in\mathcal{H}} \frac{1}{N} = 1\,,$$

and thus (5.18) is a valid prior probability distribution on the set of circadian specialists. Indeed, probabilistically (5.18) just corresponds to an assumption of independence between policies and circadian patterns.

For $h \in \mathcal{P}$ we then have $w^1_{(h,\boldsymbol{c}^h)} = f(\boldsymbol{c}^h)/N$ which when substituted

into (5.9) gives

$$\bar{C} = M \ln \frac{N}{M} + \sum_{h \in \mathcal{P}} - \ln \left(f(\boldsymbol{c}^h) \right) . \tag{5.19}$$

Our bound now depends entirely on our choice of prior, $f(\boldsymbol{\gamma})$, over circadian patterns, $\boldsymbol{\gamma} \in \{0,1\}^T$. In the following sections, we will therefore focus on two elements of our problem. The first will be on choosing a suitable prior over circadian patterns for the problem of adaptive LTM, such that we achieve a good regret bound. The second will be on how our choice can be combined with Algorithm 6 to give an efficient algorithm.

5.4.3 Implementing POLICY SPECIALISTS

Before we make explicit the chosen functional form of $f(\boldsymbol{\gamma})$ that will determine our initial weighting function $\boldsymbol{w}^1$, we will begin to consider how POLICY SPECIALISTS can be implemented with the definition of $\boldsymbol{w}^1$ given in (5.18).

For simplicity, for each policy $h \in \mathcal{H}$ and circadian $\boldsymbol{\gamma} \in \{0,1\}^T$ we introduce the weight $v_h^t(\boldsymbol{\gamma})$ for $t \in [T]$ defined as follows. For $t = 1$ we set $v_h^1 := f(\boldsymbol{\gamma})$, for all $t \in [T]$ we have

$$v_h^{t+1}(\boldsymbol{\gamma}) := \begin{cases} v_h^t(\boldsymbol{\gamma}) & \gamma_t = 0, \\ \psi_h^t v_h^t(\boldsymbol{\gamma}) & \gamma_t = 1. \end{cases}$$

It is trivial to show that for all $t \in [T]$,

$$w_{(h,\boldsymbol{\gamma})}^t = \frac{1}{N} v_h^t(\boldsymbol{\gamma}) , \tag{5.20}$$

for all $(h, \boldsymbol{\gamma}) \in \mathcal{S}$. The introduction of $v_h^t(\boldsymbol{\gamma})$ is for the sake of clarity as we will focus on the "weight" of a circadian pattern (for a given policy $h \in \mathcal{H}$) in isolation rather than the weight $w_{(h,\boldsymbol{\gamma})}^t$.

For all trials $t \in [T]$ and policies $h \in \mathcal{H}$ we now define

$$\sigma_h^t := \sum_{\boldsymbol{\gamma} \in \{0,1\}^T} [\![\gamma_t = 1]\!] v_h^t(\boldsymbol{\gamma}) ,$$

Algorithm 7 Policy Specialists with Sub-Routines

Input: $\eta > 0$

for $h \in \mathcal{H}$ **do**
  initialize(h)
end for
for $t = 1, \ldots, \mathrm{T}$ **do**
  receive $\boldsymbol{x}^t \in \mathcal{X}$
  for $h \in \mathcal{H}$ **do**
    $(\sigma_h^t, \tilde{\sigma}_h^t) \leftarrow \mathbf{get}(t, h)$
    $\epsilon_h^t \leftarrow \frac{1}{N}\sigma_h^t$
  end for
  $z_t \leftarrow \sum_{h \in \mathcal{H}} \epsilon_h^t$
  for $a \in [A]$ **do**
    $p_a^t \leftarrow \sum_{h \in \mathcal{H}} \frac{\epsilon_h^t h(\boldsymbol{x}^t)_a}{z_t}$
  end for
  select $a^t \sim \boldsymbol{p}^t$

▷ `Full Information`

  receive $\boldsymbol{\ell}^t \in [0,1]^A$
  for $h \in \mathcal{H}$ **do**
    $\hat{\ell}_h^t \leftarrow \sum_{a \in [A]} \ell_a^t h(\boldsymbol{x}^t)_a$
  end for

▷ `Partial Information`

  receive $\ell_{a^t}^t \in [0,1]$
  for $h \in \mathcal{H}$ **do**
    $\hat{\ell}_h^t \leftarrow \frac{\ell_{a^t}^t h(\boldsymbol{x}^t)_{a^t}}{p_{a^t}^t}$
  end for

  $\hat{z}^t \leftarrow \sum_{h \in \mathcal{H}} \epsilon_h^t e^{-\eta \hat{\ell}_h^t}$
  for $h \in \mathcal{H}$ **do**
    $\psi_h^t \leftarrow \frac{z_t e^{-\eta \hat{\ell}_h^t}}{\hat{z}^t}$
    update(t, h)
  end for
end for

and

$$\tilde{\sigma}_h^t := \sum_{\boldsymbol{\gamma} \in \{0,1\}^T} [\![\gamma_t = 0]\!] v_h^t(\boldsymbol{\gamma})\,.$$

Note now that we have

$$\epsilon_h^t = \sum_{\boldsymbol{\gamma} \in \{0,1\}^T} [\![\gamma_t = 1]\!] w_{(h,\boldsymbol{\gamma})}^t = \frac{1}{N}\sigma_h^t\,,$$

Routine 1 initialize(h)

1: **for** $\boldsymbol{\gamma} \in \{0,1\}^T$ **do**
2: $\quad v_h^1(\boldsymbol{\gamma}) \leftarrow f(\boldsymbol{\gamma})$
3: **end for**

Routine 2 get(t,h)

1: **return** $\left(\sum_{\boldsymbol{\gamma}\in\{0,1\}^T} [\![\gamma_t = 1]\!] v_h^t(\boldsymbol{\gamma}),\ \sum_{\boldsymbol{\gamma}\in\{0,1\}^T} [\![\gamma_t = 0]\!] v_h^t(\boldsymbol{\gamma})\right)$

Routine 3 update(t,h)

1: **for** $\boldsymbol{\gamma} \in \{0,1\}^T$ **do**
2: $\quad v_h^{t+1}(\boldsymbol{\gamma}) \leftarrow \begin{cases} v_h^t(\boldsymbol{\gamma}), & \gamma_t = 0 \\ \psi_h^t v_h^t(\boldsymbol{\gamma}), & \gamma_t = 1 \end{cases}$
3: **end for**

and thus on each trial, knowledge of σ_h^t is all that is required to compute ϵ_h^t, which is needed in Algorithm 6 (line 4). Later, our algorithm will also require $\tilde{\sigma}_h^t$, and so we will now re-write Algorithm 6 by introducing three sub-routines, as given in Algorithm 7. We name these routines **initialize**, **get**, and **update**, which are called in Algorithm 7 on lines 2, 7, and 22 respectively. The **get** method returns both σ_h^t and $\tilde{\sigma}_h^t$. The algorithm now only takes as input the learning rate, η (and implicitly a choice of prior $f(\boldsymbol{\gamma})$). The abstraction of these routines allows us to assume (for the time being) the existence of an oracle which we can call on each trial for any particular choice of $f(\boldsymbol{\gamma})$, simplifying our presentation and analysis. We will then explicitly define these routines for our particular choice of $f(\boldsymbol{\gamma})$ in Section 5.6.2, removing the need for such an oracle.

In their current form, the required methods of **initialize**, **get**, and **update** are shown in routines 1, 2, and 3 respectively. Observe that Algorithm 7 with these routines recovers Algorithm 6 exactly. Note that all three of these routines apply to a unique policy, h, independently of all other policies. Of course, to compute these routines explicitly would require a per-trial time complexity of $\mathcal{O}(2^T)$ per policy. In the next section we introduce our method of choosing

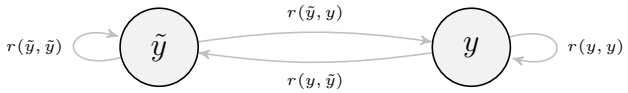


Figure 5.2: The CGMC of [2] for the problem of switching with (non-adaptive) memory. Observe that there are only two states - one awake and one asleep, i.e., $\mathcal{W} = \{y\}$ and $\widetilde{\mathcal{W}} = \{\tilde{y}\}$. Furthermore, the transition function $r_t(u, y)$ for $u, y \in \mathcal{Y}$ is constant for all $t \in \mathbb{N}$.

a prior weight, $f(\boldsymbol{\gamma})$, over circadian patterns, generalising the method of [2], which used a simple Markov chain.

5.5 Circadian-Generating Markov Chains

We have shown that Algorithm 6 (or equivalently Algorithm 7 with routines 1, 2, and 3) has a regret bound $\mathcal{R}(\boldsymbol{h}) \leq \mathcal{O}(\sqrt{A^\dagger \bar{C} T})$, with $\bar{C}$ given in (5.19). We have seen, however, that implementing such a specialist algorithm explicitly requires exponential time and space per trial. Our goal now, then, is to choose a prior function $f(\boldsymbol{\gamma})$ over circadian patterns for our problem of adaptive LTM that will lead to a good regret bound, and a set of routines for Algorithm 7 that can be implemented efficiently. In the following sections we show why our choice of initial weighting is well-suited to our problem and develop these three routines. Recall that although we develop our algorithm as if we had prior knowledge of the number of trials, T, the algorithm does not need this knowledge a-priori.

Our method primarily extends the work of [2, 12], which both use simple Markov chain priors over circadian specialists to achieve bounds for switching with memory. We generalize this idea by introducing the concept of a circadian-generating Markov chain (CGMC).

A CGMC is a quadruple $(\mathcal{Y}, \mathcal{W}, \iota, \boldsymbol{r})$, where $\mathcal{Y}$ is a countable set of "states." We describe the subset $\mathcal{W} \subseteq \mathcal{Y}$ of states as *awake* states, and the complement of this set, $\widetilde{\mathcal{W}} := \mathcal{Y} \setminus \mathcal{W}$ as the set of *asleep* states. We define $\iota : \mathcal{Y} \to [0, 1]$ as

the "initial state function" such that

$$\sum_{y \in \mathcal{Y}} \iota(y) = 1\,, \tag{5.21}$$

and finally $\boldsymbol{r} : \mathcal{Y} \times \mathcal{Y} \to [0,1]^{\infty}$ is a transition function such that for all $(u,t) \in \mathcal{Y} \times \mathbb{N}$,

$$\sum_{y \in \mathcal{Y}} r_t(u,y) = 1\,. \tag{5.22}$$

Note that $\boldsymbol{r}$ is a vector-valued function whose range is infinite-dimensional, and $r_t(u,y)$ denotes the transition probability from state u to y on trial t.

The novelty of this definition of a CGMC over previous works is two-fold. First, note that we allow for multiple states to be "awake" ($|\mathcal{W}| \geq 1$) or "asleep" ($|\widetilde{\mathcal{W}}| \geq 1$), whereas, for example, the Markov chain in [2] has only single awake and asleep states (see Figure 5.2). Secondly, our transition function $\boldsymbol{r}$ is not static; instead, the transition probability between states is allowed to vary over time.

We will briefly consider the required behavior of our CGMC for the problem of adaptive LTM, comparing it to the simple CGMC given in Figure 5.2, and why our generalized notion of a CGMC is necessary. For the problem of adaptive LTM, our inductive bias will be to favor *shorter* periods of waking/sleeping relative to the length of the entire trial sequence, as well as (more importantly) long periods of being asleep. For a given policy $h \in \mathcal{H}$ and epoch i, if $h \in \mathcal{P}^i$ then during that epoch we expect h to be the relevant policy for some segments in that epoch, and irrelevant for other segments, this intuitively corresponds to a circadian pattern waking up and falling asleep frequently during that epoch, relative to the entire trial sequence. If $h \notin \mathcal{P}^i$ then this requires a circadian which is asleep for the entire epoch duration (see Figure 5.3). Thus the circadian patterns of interest are those that have periods of waking up/falling asleep frequently, followed by long periods of "deep sleep." Our CGMC will allow us to capture this notion of deep sleep by having multiple awake states and multiple asleep states.

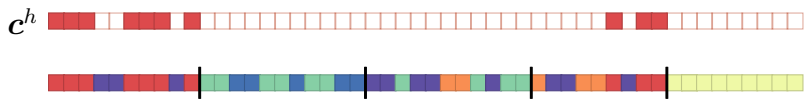


Figure 5.3: Our toy example trial sequence from Figure 5.1, with the circadian $\boldsymbol{\gamma} = \boldsymbol{c}^h$ for the "red" policy h shown above. A filled square implies $\gamma_t = 1$ and an empty square impiles $\gamma_t = 0$. Observe that in the adaptive LTM model the circadian patterns of interest have periods of waking up/falling asleep frequently, followed by long periods of "deep sleep."

We now consider the simple Markov prior induced by the CGMC in Figure 5.2 and how it is ill-suited to the problem of adaptive LTM. Observe that the constant transition probabilities of that CGMC mean that regardless of how long the Markov chain has been in the sleep state, it always has the same probability of waking up (and vice versa). In our model, however, the longer the Markov chain is in an asleep state, the less probability it should have of waking up, as intuitively this corresponds to a policy being less likely to be in the current small pool $\mathcal{P}^i$, and irrelevant for possibly a long time, as shown in Figure 5.3. Indeed, comparing Figure 5.3 with Figure 4.1, the difference in the characteristics of the desired circadian patterns becomes clear. The same behavior is also true with the reversed notions of awake and asleep. Our CGMC will achieve this behavior by encouraging periods of waking up/falling asleep relatively frequently and, additionally, periods of prolonged sleep as required. Practically, this will be achieved by having multiple awake (and asleep) states in our CGMC being associated with different *levels* of "wakefulness." Intuitively, our transition probabilities will be designed such that the more "wide awake" or "deeply asleep" a given state is, the *harder* it will be to fall asleep or wake up respectively.

Given a CGMC $(\mathcal{Y}, \mathcal{W}, \iota, \boldsymbol{r})$, our prior function $f(\boldsymbol{\gamma})$ over circadian patterns is defined as follows. For all $T \in \mathbb{N}$ and $\boldsymbol{z} \in \mathcal{Y}^T$ let

$$\hat{w}(\boldsymbol{z}) := \iota(z_1) \prod_{t=1}^{T-1} r_t(z_t, z_{t+1}) \,, \tag{5.23}$$

be the probability of the sequence $\boldsymbol{z}$. For any $T \in \mathbb{N}$ and $\boldsymbol{\gamma} \in \{0,1\}^T$, we define

$$\Gamma(\boldsymbol{\gamma}) := \{\boldsymbol{z} \in \mathcal{Y}^T \mid \forall t \in [T], \gamma_t = [\![z_t \in \mathcal{W}]\!]\} \tag{5.24}$$

to be the set of sequences of states $\boldsymbol{z}$ of length T which correspond to a circadian pattern $\boldsymbol{\gamma}$. Observe that since we now have multiple distinct states which are considered awake or asleep, then different sequences of states $\boldsymbol{z} \in \mathcal{Y}^T$ may "agree" on the same circadian pattern $\boldsymbol{\gamma} \in \{0,1\}^T$. We finally define the function $f(\boldsymbol{\gamma})$ by

$$f(\boldsymbol{\gamma}) := \sum_{\boldsymbol{z} \in \Gamma(\boldsymbol{\gamma})} \hat{w}(\boldsymbol{z}) \tag{5.25}$$

for all $\boldsymbol{\gamma} \in \{0,1\}^T$.

In the following Proposition we show that $f(\boldsymbol{\gamma})$ is a valid distribution on circadian patterns.

Proposition 27. *The function $f(\boldsymbol{\gamma})$ defined in* (5.25) *is such that*

$$\sum_{\boldsymbol{\gamma} \in \{0,1\}^T} f(\boldsymbol{\gamma}) = 1\,.$$

Proof. Note that for all $t \in [T]$ and $\boldsymbol{z} \in \mathcal{Y}^t$, it follows from (5.23) and (5.22) that

$$\begin{aligned}
\sum_{\boldsymbol{u} \in \mathcal{Y}^{t+1} : \boldsymbol{u}^{|t} = \boldsymbol{z}} \hat{w}(\boldsymbol{u}) &= \sum_{\boldsymbol{u} \in \mathcal{Y}^{t+1} : \boldsymbol{u}^{|t} = \boldsymbol{z}} \iota(u_1) \prod_{s=1}^{t} r_s(u_s, u_{s+1}) \\
&= \iota(z_1) \left(\prod_{s=1}^{t-1} r_s(z_s, z_{s+1}) \right) \sum_{u_{t+1} \in \mathcal{Y}} r_t(z_t, u_{t+1}) \\
&= \hat{w}(\boldsymbol{z})\,,
\end{aligned}$$

which, by reverse induction on t gives, for all $t \in [T]$ and $\boldsymbol{z} \in \mathcal{Y}^t$,

$$\hat{w}(\boldsymbol{z}) = \sum_{\boldsymbol{u} \in \mathcal{Y}^T : \boldsymbol{u}^{|t} = \boldsymbol{z}} \hat{w}(\boldsymbol{u})\,. \tag{5.26}$$

Noting that $\{\Gamma(\boldsymbol{\gamma}) \mid \boldsymbol{\gamma} \in \{0,1\}^T\}$ partitions $\mathcal{Y}^T$ we then have

$$\sum_{\boldsymbol{\gamma}\in\{0,1\}^T} f(\boldsymbol{\gamma}) = \sum_{\boldsymbol{\gamma}\in\{0,1\}^T} \sum_{\boldsymbol{z}\in\Gamma(\boldsymbol{\gamma})} \hat{w}(\boldsymbol{z}) = \sum_{\boldsymbol{z}\in\mathcal{Y}^T} \hat{w}(\boldsymbol{z})\,,$$

which when combined with (5.21) and (5.26) gives

$$\begin{aligned}
\sum_{\boldsymbol{\gamma}\in\{0,1\}^T} f(\boldsymbol{\gamma}) &= \sum_{\boldsymbol{u}\in\mathcal{Y}^T} \hat{w}(\boldsymbol{u}) \\
&= \sum_{\boldsymbol{z}\in\mathcal{Y}^1} \sum_{\boldsymbol{u}\in\mathcal{Y}^T:u_1=z_1} \hat{w}(\boldsymbol{u}) \\
&= \sum_{\boldsymbol{z}\in\mathcal{Y}^1} \hat{w}(\boldsymbol{z}) \\
&= \sum_{z\in\mathcal{Y}} \iota(z) \\
&= 1\,,
\end{aligned}$$

as required. □

5.5.1 Implementing a CGMC

Before we explore CGMCs further and define our own for the problem of adaptive LTM, we will consider how we might implement the three required routines for Algorithm 7 now that $f(\boldsymbol{\gamma})$ is defined as in (5.25).

Firstly for convenience, for all $t \in [T]$ and $x \in \{0,1\}$ let

$$\hat{\psi}_h^t(x) := \begin{cases} 1 & x = 0, \\ \psi_h^t & x = 1, \end{cases}$$

such that for all $t \in [T]$ and $\boldsymbol{\gamma} \in \{0,1\}^T$ we have

$$v_h^t(\boldsymbol{\gamma}) = f(\boldsymbol{\gamma}) \prod_{s=1}^{t-1} \hat{\psi}_h^s(\gamma_s)\,. \tag{5.27}$$

For all $h \in \mathcal{H}$, for all $t \in [T]$ and $\boldsymbol{z} \in \mathcal{Y}^t$ we also define

$$\zeta(h, \boldsymbol{z}) := \hat{w}(\boldsymbol{z}) \prod_{s=1}^{t-1} \hat{\psi}_h^s([\![z_s \in \mathcal{W}]\!]) \,.$$

This quantity is analogous to (5.27), but we are now considering the more general state sequences $\boldsymbol{z} \in \mathcal{Y}^T$ rather than circadian patterns $\boldsymbol{\gamma} \in \{0,1\}^T$. Indeed, we now show how we can compute $\sigma_h^t = \sum_{\boldsymbol{\gamma} \in \{0,1\}^T} [\![\gamma_t = 1]\!] v_h^t(\boldsymbol{\gamma})$ in terms of $\zeta(h, \boldsymbol{z})$.

Note that for all $t \in [T]$ and $\boldsymbol{\delta} \in \{0,1\}^t$ we have

$$\bigcup \{\Gamma(\boldsymbol{\gamma}) \mid \boldsymbol{\gamma} \in \{0,1\}^T \wedge \boldsymbol{\gamma}^{|t} = \boldsymbol{\delta}\} = \bigcup \{\{\boldsymbol{u} \in \mathcal{Y}^T \mid \boldsymbol{u}^{|t} = \boldsymbol{z}\} \mid \boldsymbol{z} \in \Gamma(\boldsymbol{\delta})\} \,,$$

so

$$\begin{aligned} \sum_{\boldsymbol{\gamma} \in \{0,1\}^T : \boldsymbol{\gamma}^{|t} = \boldsymbol{\delta}} f(\boldsymbol{\gamma}) &= \sum_{\boldsymbol{\gamma} \in \{0,1\}^T : \boldsymbol{\gamma}^{|t} = \boldsymbol{\delta}} \left(\sum_{\boldsymbol{z} \in \Gamma(\boldsymbol{\gamma})} \hat{w}(\boldsymbol{z}) \right) \\ &= \sum_{\boldsymbol{z} \in \Gamma(\boldsymbol{\delta})} \left(\sum_{\boldsymbol{u} \in \mathcal{Y}^T : \boldsymbol{u}^{|t} = \boldsymbol{z}} \hat{w}(\boldsymbol{u}) \right) , \end{aligned}$$

which by (5.26) is equal to $\sum_{\boldsymbol{z} \in \Gamma(\boldsymbol{\delta})} \hat{w}(\boldsymbol{z})$. Hence by (5.27), we have

$$\begin{aligned} \sum_{\boldsymbol{\gamma} \in \{0,1\}^T : \boldsymbol{\gamma}^{|t} = \boldsymbol{\delta}} v_h^t(\boldsymbol{\gamma}) &= \sum_{\boldsymbol{\gamma} \in \{0,1\}^T : \boldsymbol{\gamma}^{|t} = \boldsymbol{\delta}} f(\boldsymbol{\gamma}) \prod_{s=1}^{t-1} \hat{\psi}_h^s(\delta_s) \\ &= \sum_{\boldsymbol{z} \in \Gamma(\boldsymbol{\delta})} \hat{w}(\boldsymbol{z}) \prod_{s=1}^{t-1} \hat{\psi}_h^s(\delta_s) \,. \end{aligned} \tag{5.28}$$

Recalling that $\sigma_h^t = \sum_{\boldsymbol{\gamma} \in \{0,1\}^T} [\![\gamma_t = 1]\!] v_h^t(\boldsymbol{\gamma})$, note that

$$\sum_{\boldsymbol{\gamma} \in \{0,1\}^T} [\![\gamma_t = 1]\!] v_h^t(\boldsymbol{\gamma}) = \sum_{\boldsymbol{\delta} \in \{0,1\}^t : \delta_t = 1} \left(\sum_{\boldsymbol{\gamma} \in \{0,1\}^T : \boldsymbol{\gamma}^{|t} = \boldsymbol{\delta}} v_h^t(\boldsymbol{\gamma}) \right) .$$

Combining this with (5.28) then gives

$$\begin{aligned}
\sigma_h^t &= \sum_{\boldsymbol{\gamma}\in\{0,1\}^T} [\![\gamma_t = 1]\!] v_h^t(\boldsymbol{\gamma}) \\
&= \sum_{\boldsymbol{\delta}\in\{0,1\}^t:\delta_t=1} \left(\sum_{\boldsymbol{\gamma}\in\{0,1\}^T:\boldsymbol{\gamma}^{|t}=\boldsymbol{\delta}} v_h^t(\boldsymbol{\gamma}) \right) \\
&= \sum_{\boldsymbol{\delta}\in\{0,1\}^t:\delta_t=1} \left(\sum_{\boldsymbol{z}\in\Gamma(\boldsymbol{\delta})} \hat{w}(\boldsymbol{z}) \prod_{s=1}^{t-1} \hat{\psi}_h^s(\delta_s) \right) \\
&= \sum_{\boldsymbol{\delta}\in\{0,1\}^t:\delta_t=1} \left(\sum_{\boldsymbol{z}\in\Gamma(\boldsymbol{\delta})} \zeta(h,\boldsymbol{z}) \right) \\
&= \sum_{\boldsymbol{z}\in\mathcal{Y}^t} [\![z_t \in \mathcal{W}]\!] \zeta(h,\boldsymbol{z}) \,. \qquad (5.29)
\end{aligned}$$

A similar argument can be used to show that

$$\tilde{\sigma}_h^t = \sum_{\boldsymbol{z}\in\mathcal{Y}^t} [\![z_t \in \widetilde{\mathcal{W}}]\!] \zeta(h,\boldsymbol{z}) \,.$$

We have thus expressed the quantities σ_h^t and $\tilde{\sigma}_h^t$ as sums over sequences of states $\boldsymbol{z} \in \mathcal{Y}^T$ rather than sums over circadian patterns $\boldsymbol{\gamma} \in \{0,1\}^T$. Intuitively, the set of circadian patterns defining our set of specialists now correspond to a (possibly larger) set of sequences of states in $\mathcal{Y}$ rather than sequences made of elements of $\{0,1\}$ directly. Recall that these "state sequences", however, get mapped to a circadian pattern of awake/asleep states.

We now define, for $u \in \mathcal{Y}$,

$$\mu_h^t(u) := \sum_{\boldsymbol{z}\in\mathcal{Y}^t:z_t=u} \zeta(h,\boldsymbol{z}) \,,$$

which can be seen as the weight of all "state-sequences" (associated with policy h) on trial t whose t^{th} state is u. This then gives

$$\sigma_h^t = \sum_{u\in\mathcal{W}} \mu_h^t(u) \qquad \text{and} \qquad \tilde{\sigma}_h^t = \sum_{u\in\widetilde{\mathcal{W}}} \mu_h^t(u) \,, \qquad (5.30)$$

for the quantities required by our **get** method.

Note that this reduces the computation of the quantities σ_h^t and $\tilde{\sigma}_h^t$ to sums over states $u \in \mathcal{W}$ and $u \in \widetilde{\mathcal{W}}$ only, although we have not yet shown how we incrementally update μ_h^t on each trial. We will do that now. Note that we initialize with

$$\mu_h^1(y) = \hat{w}(y) = \iota(y)\,, \tag{5.31}$$

for all $y \in \mathcal{Y}$ and $h \in \mathcal{H}$.

In order to update μ_h^t, first note that for all $t \in [T]$, and $\boldsymbol{z} \in \mathcal{Y}^{t+1}$ we have

$$\begin{aligned}
\zeta(h, \boldsymbol{z}) &= \left(\iota(z_1)\prod_{s=1}^{t} r_s(z_s, z_{s+1})\right)\left(\prod_{s=1}^{t}\hat{\psi}_h^s([\![z_s \in \mathcal{W}]\!])\right) \\
&= \iota(z_1)\prod_{s=1}^{t} r_s(z_s, z_{s+1})\hat{\psi}_h^s([\![z_s \in \mathcal{W}]\!]) \\
&= \zeta(h, \boldsymbol{z}^{|t})\hat{\psi}_h^t([\![z_t \in \mathcal{W}]\!])r_t(z_t, z_{t+1})
\end{aligned}$$

and hence for all $u, u' \in \mathcal{Y}$ we have

$$\sum_{\boldsymbol{z}\in\mathcal{Y}^{t+1}:(z_t=u)\wedge(z_{t+1}=u')} \zeta(h, \boldsymbol{z}) = \sum_{\boldsymbol{z}\in\mathcal{Y}^{t}:z_t=u} \zeta(h, \boldsymbol{z})\hat{\psi}_h^t([\![u \in \mathcal{W}]\!])r_t(u, u'),$$

so for all $u' \in \mathcal{Y}$ we have

$$\begin{aligned}
\mu_h^{t+1}(u') &= \sum_{\boldsymbol{z}\in\mathcal{Y}^{t+1}:z_{t+1}=u'} \zeta(h, \boldsymbol{z}) \\
&= \sum_{u\in\mathcal{Y}}\left(\sum_{\boldsymbol{z}\in\mathcal{Y}^{t+1}:(z_t=u)\wedge(z_{t+1}=u')} \zeta(h, \boldsymbol{z})\right) \\
&= \sum_{u\in\mathcal{Y}} \mu_h^t(u)\hat{\psi}_h^t([\![u \in \mathcal{W}]\!])r_t(u, u') \\
&= \psi_h^t \sum_{u\in\mathcal{W}} \mu_h^t(u)r_t(u, u') + \sum_{u\in\widetilde{\mathcal{W}}} \mu_h^t(u)r_t(u, u')\,,
\end{aligned} \tag{5.32}$$

as required. With this update our three routines, **initialize**, **get**, and **update** can now be expressed in the form given in routines 4, 5, and 6 respectively.

Routine 4 initialize(h)

1: **for** $u \in \mathcal{Y}$ **do**
2: $\quad \mu_h^1(u) \leftarrow \iota(u)$
3: **end for**

Routine 5 get(t, h)

1: **return** $\left(\sum_{u \in \mathcal{W}} \mu_h^t(u), \sum_{u \in \widetilde{\mathcal{W}}} \mu_h^t(u) \right)$

Routine 6 update(t, h)

1: **for** $u' \in \mathcal{Y}$ **do**
2: $\quad \mu_h^{t+1}(u') \leftarrow \psi_h^t \sum_{u \in \mathcal{W}} \mu_h^t(u) r_t(u, u') + \sum_{u \in \widetilde{\mathcal{W}}} \mu_h^t(u) r_t(u, u')$
3: **end for**

So far we have reduced the computation of σ_h^t and $\tilde{\sigma}_h^t$ from a sum over $\boldsymbol{\gamma} \in \{0,1\}^T$ requiring $\mathcal{O}(2^T)$ time per trial, to a sum over $u \in \mathcal{Y}$. Additionally our **update** method now requires for each state $u' \in \mathcal{Y}$, a sum over all states $u \in \mathcal{Y}$, requiring $\mathcal{O}(|\mathcal{Y}|^2)$ time per trial (per policy).

Recall the previous work [2] (the circadian specialists algorithm with the simple Markov chain prior generated by the CGMC shown in Figure 5.2) which we studied in Chapter 4, where $\mathcal{W} = \{a\}$ and $\widetilde{\mathcal{W}} = \{s\}$). We can now see that with this CGMC, routine 6 gives the update of that algorithm with

$$\begin{aligned} \mu_h^{t+1}(a) &= \psi_h^t \sum_{u \in \mathcal{W}} \mu_h^t(u) r_t(u, a) + \sum_{u \in \widetilde{\mathcal{W}}} \mu_h^t(u) r_t(u, a) \\ &= \psi_h^t \mu_h^t(a) r_t(a, a) + \mu_h^t(s) r_t(s, a)\,, \end{aligned}$$

and

$$\begin{aligned} \mu_h^{t+1}(s) &= \psi_h^t \sum_{u \in \mathcal{W}} \mu_h^t(u) r_t(u, s) + \sum_{u \in \widetilde{\mathcal{W}}} \mu_h^t(u) r_t(u, s) \\ &= \psi_h^t \mu_h^t(a) r_t(a, s) + \mu_h^t(s) r_t(s, s)\,, \end{aligned}$$

which correspond with the updates (4.33) and (4.34) respectively.

This update is efficient for a two-state Markov chain (as in Figure 5.2). In our model, however, we have an extended set of states, $\mathcal{Y}$, and the time complexity of our update scales with $\mathcal{O}(|\mathcal{Y}|^2)$. Thus in order to reduce the time complexity of this update further, we will require some structure to our transition function $\boldsymbol{r}$ which can be exploited. In the next section we consider such a structure.

5.5.2 A CGMC for Adaptive LTM - Intuition

In this section we briefly sketch out the form of our approach and develop the intuition behind it. We do this by first describing a more simple (although inefficient) approach, and then describe how our algorithm corresponds, in a rough sense, to an efficient implementation of this approach. The subsequent sections then develop the algorithm.

We first sketch how our approach will lead to the regret bound stated in Theorem 26. Observe that (5.19) shows that our regret bound depends entirely on our choice of prior distribution, $f(\boldsymbol{\gamma})$, over circadian patterns. We now sketch our chosen distribution over circadian patterns for the problem of adaptive LTM. Note that a distribution over infinite sequences $\{0,1\}^*$ can be specified by the lengths, λ, of its contiguous segments of zeroes and ones

$$\underbrace{00\ldots0}_{\tilde{\lambda}_1}\underbrace{11\ldots1}_{\lambda_1}\underbrace{00\ldots0}_{\tilde{\lambda}_2}\underbrace{11\ldots1}_{\lambda_2}\ldots\ldots$$

such that any length-T prefix $\boldsymbol{\gamma} \in \{0,1\}^T$ of an infinite sequence may always be described by the set of length-pairs $\mathcal{J}_{\boldsymbol{\gamma}} := \{(\tilde{\lambda}_1, \lambda_1), \ldots, (\tilde{\lambda}_J, \lambda_J)\}$ where for notational convenience we may define $\tilde{\lambda}_1 := 0$ and/or $\lambda_J := 0$ (if $\gamma_1 = 1$ and/or $\gamma_T = 0$, respectively) as necessary such that $\boldsymbol{\gamma}$ consists of $2J$ contiguous segments. To achieve a good regret bound we will observe that we will require that the length of each contiguous segment in $\boldsymbol{\gamma}$ is independent of the previous segments and that these lengths follow a quadratic power law such that the

following is satisfied:

$$f(\boldsymbol{\gamma}) \geq \frac{1}{2(\tilde{\lambda}_1+1)^2(\lambda_1+1)^2\ldots(\tilde{\lambda}_J+1)^2(\lambda_J+1)^2} = \frac{1}{2}\prod_{(\tilde{\lambda}_j,\lambda_j)\in\mathcal{J}_{\boldsymbol{\gamma}}}\frac{1}{(\tilde{\lambda}_j+1)^2(\lambda_j+1)^2}\,. \tag{5.33}$$

Substituting (5.33) into (5.19) then gives a regret bound of $\mathcal{R}(\boldsymbol{h}) \leq \mathcal{O}(\sqrt{A^\dagger CT})$ with

$$C := M\ln\left(\frac{N}{M}\right) + \sum_{h\in\mathcal{P}}\sum_{(\tilde{\lambda}_j,\lambda_j)\in\mathcal{J}_{\boldsymbol{c}^h}}\Big(\ln(\tilde{\lambda}_j)+\ln(\lambda_j)\Big)\,, \tag{5.34}$$

and thus in our bound we would pay (up to constant factors) the logarithm of the *length* of each sleep/wake segment of $\boldsymbol{\gamma}$ for each of our M specialists required to predict in accordance with the sequence $\boldsymbol{h}$.

We now sketch how the bound (5.34) obtains the bound given in Theorem 26 in terms of our adaptive LTM model. Without loss of generality, assume that each epoch starts on a switch. For some epoch $i \in [E]$ let $\mathcal{J}^i_{\boldsymbol{c}^h} := \{(\tilde{\lambda}_k,\lambda_k),\ldots,(\tilde{\lambda}_{k+J^i-1},\lambda_{k+J^i-1})\}$ for some $k \geq 1$ be the set of J^i length-pairs of contiguous segments in $\boldsymbol{\gamma}$ fully contained within epoch i. Observe that we have the following

$$\sum_{h\in\mathcal{P}^i}\sum_{(\tilde{\lambda}_j,\lambda_j)\in\mathcal{J}^i_{\boldsymbol{c}^h}}(\tilde{\lambda}_j+\lambda_j) \leq T^i + (M^i-1)T^i = M^iT^i\,. \tag{5.35}$$

Since there are at most $2K^i + M^i$ terms being summed on the left-hand side of (5.35), then using Jensen's inequality gives[1]

$$\sum_{h\in\mathcal{P}^i}\sum_{(\tilde{\lambda}_j,\lambda_j)\in\mathcal{J}^i_{\boldsymbol{c}^h}}\Big(\ln(\tilde{\lambda}_j)+\ln(\lambda_j)\Big) \leq (2K^i+M^i)\ln\left(\frac{M^iT^i}{2K^i+M^i}\right) \in \mathcal{O}\left(K^i\ln\left(\frac{M^iT^i}{K^i}\right)\right),$$

since $K^i \geq M^i$. For the remaining "long sleeping" segments of our circadian

[1]That for a set $\{x_i \mid i\in[k]\} \subseteq \mathbb{N}$, we have $\sum_{i\in[k]}\ln x_i \leq k\ln(\sum_{i\in[k]}x_i/k)$.

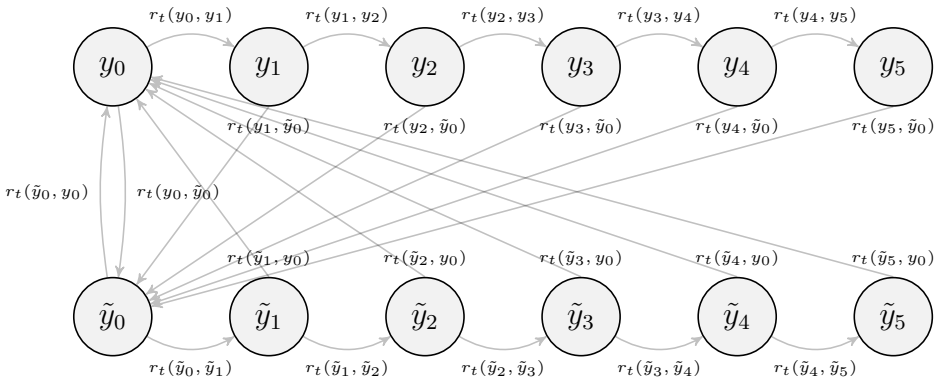


Figure 5.4: A finite example of a simple (but "inefficient") circadian-generating Markov chain, $\mathcal{M}_A$, for adaptive LTM.

patterns not fully contained within a given epoch (i.e., those corresponding to $h \in \mathcal{P}$ such that $h \notin \mathcal{P}^i$ for some epoch $i \in [E]$) we have $\mathcal{O}(\Phi)$ such segments across the entire trial sequence whose summed lengths is no more than MT. A similar application of Jensen's inequality and summing across all epochs therefore gives

$$\sum_{h \in \mathcal{P}} \sum_{(\tilde{\lambda}_j, \lambda_j) \in \mathcal{J}_{\mathbf{c}^h}} \left(\ln\left(\tilde{\lambda}_j\right) + \ln\left(\lambda_j\right) \right) \in \mathcal{O}\left(\sum_{i=1}^{E} K^i \ln\left(\frac{M^i T^i}{K^i} \right) + \Phi \ln\left(\frac{MT}{\Phi} \right) \right).$$

We now describe a simple Markov chain which induces a distribution $f(\boldsymbol{\gamma})$ over circadian patterns which satisfies (5.33) and thus when "implemented" achieves the regret bound of Theorem 26. This Markov chain, which we denote $\mathcal{M}_A := (\mathcal{Y}, \mathcal{W}, \iota, r)$ is simple in that the transition probabilities are time-homogenous, that is r is fixed and is not required to be a vector-valued function.

We denote our set of awake states, $\mathcal{W}$, by $\mathcal{W} := \{y_q \mid q \in \mathbb{N} \cup \{0\}\}$ and our set of asleep states, $\widetilde{\mathcal{W}}$, by $\widetilde{\mathcal{W}} := \{\tilde{y}_q \mid q \in \mathbb{N} \cup \{0\}\}$. We can thus interpret the different states $y_0, y_1, \ldots$ in $\mathcal{W}$ as different levels of "wakefulness," where state y_{q+1} is interpreted as being more *wide awake* than state y_q. Similarly state $\tilde{y}_{q+1}$ is interpreted as being more *deeply asleep* than state $\tilde{y}_q$. Intuitively, being more awake (or more asleep) will mean that it is *harder* (less likely) to fall asleep (or wake up).

The graphical structure of $\mathcal{M}_A$ is shown in Fig. 5.4. Observe that waking up (falling asleep) corresponds to transitioning to state y_0 ($\tilde{y}_0$), and that staying awake (staying asleep) corresponds to transitioning from some state y_q to y_{q+1} ($\tilde{y}_q$ to $\tilde{y}_{q+1}$) respectively for some $q \in \mathbb{N} \cup \{0\}$. The transition probabilities of $\mathcal{M}_A$ are as follows. We have $\iota(y_0) = \iota(\tilde{y}_0) = 1/2$ and for $q \in \mathbb{N} \cup \{0\}$,

$$r(y_q, \tilde{y}_0) := r(\tilde{y}_q, y_0) := \frac{1}{q+2}\,, \qquad r(y_q, y_{q+1}) := r(\tilde{y}_q, \tilde{y}_{q+1}) := 1 - \frac{1}{q+2}\,.$$

Observe that our different levels of wakefulness are reflected in the fact that $r(\tilde{y}_{q+1}, y_0) < r(\tilde{y}_q, y_0)$, and thus the longer a specialist is asleep, the more deeply asleep it is and waking up is less probable.

A sequence $\boldsymbol{u} \in \mathcal{Y}^T$ is associated with a circadian pattern $\boldsymbol{\gamma} \in \{0,1\}^T$ with the mapping $\gamma_t := [\![u_t \in \mathcal{W}]\!]$. Observe that a specialist sleeping for $\lambda \geq 1$ trials before waking up now corresponds to the sequence of states $(\tilde{y}_0, \ldots, \tilde{y}_{\lambda-1}, y_0)$ in $\mathcal{M}_A$, and such a sequence occurs with probability $(\prod_{q=0}^{\lambda-2} r(\tilde{y}_q, \tilde{y}_{q+1}))r(\tilde{y}_{\lambda-1}, y_0) = (\prod_{q=0}^{\lambda-2} \frac{q+1}{q+2})\frac{1}{\lambda+1} = \frac{1}{\lambda(\lambda+1)} > \frac{1}{(\lambda+1)^2}$. Thus condition (5.33) is satisfied and the regret bound of Theorem 26 holds. Note that somewhat astonishingly $\mathcal{M}_A$ achieves this while being completely (switching) parameter-free.

Observe that given the graphical structure of $\mathcal{M}_A$, then computing routines 5 and 6 would unfortunately require $\mathcal{O}(t)$ time on trial t (per policy). In the following sections we therefore introduce a modification to the structure of $\mathcal{M}_A$ which will give us an alternative Markov chain that also achieves the same regret bound while reducing our computation time drastically. This modification has two components, the first is a modification to the graphical structure of $\mathcal{M}_A$ by allowing a particle following a trajectory through the chain to remain on a given state for multiple trials. The second is to introduce a time-dependence to the transition probabilities such that the majority of the quantities summed in routines 5 and 6 may be cached, reducing our computation time to $\mathcal{O}(1)$ time per trial per policy. This is made clear in the next sections.

5.5.3 Transition Mass

To see how we can exploit any structure in our CGMC to reduce the computation times of routines 5 and 6, we will require the following definitions for convenience. We first define for all $t \in [T]$ and $h \in \mathcal{H}$,

$$\pi_h^t := \prod_{s=1}^{t-1} \psi_h^s \,.$$

Then for all $u \in \mathcal{Y}$ and $h \in \mathcal{H}$ we define

$$\hat{\mu}_h^t(u) := \begin{cases} \mu_h^t(u) & u \in \widetilde{\mathcal{W}} \\ \frac{\mu_h^t(u)}{\pi_h^t} & u \in \mathcal{W} \,, \end{cases}$$

and the function $\hat{r}_h^t : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ by

$$\hat{r}_h^t(u, u') = \hat{\mu}_h^t(u) r_t(u, u') \,.$$

We will call $\hat{r}_h^t(u, u')$ the "transition mass" from state u to state u' at time t. We now show how the quantity $\hat{\mu}_h^t$ can be updated from trial t to trial $t+1$ using this transition mass $\hat{r}_h^t$. Note that from (5.32) we have

$$\begin{aligned} \mu_h^{t+1}(u') &= \psi_h^t \sum_{u \in \mathcal{W}} \mu_h^t(u) r_t(u, u') + \sum_{u \in \widetilde{\mathcal{W}}} \mu_h^t(u) r_t(u, u') \\ &= \psi_h^t \sum_{u \in \mathcal{W}} \pi_h^t \hat{\mu}_h^t(u) r_t(u, u') + \sum_{u \in \widetilde{\mathcal{W}}} \hat{\mu}_h^t(u) r_t(u, u') \\ &= \pi_h^{t+1} \sum_{u \in \mathcal{W}} \hat{\mu}_h^t(u) r_t(u, u') + \sum_{u \in \widetilde{\mathcal{W}}} \hat{\mu}_h^t(u) r_t(u, u') \\ &= \pi_h^{t+1} \sum_{u \in \mathcal{W}} \hat{r}_h^t(u, u') + \sum_{u \in \widetilde{\mathcal{W}}} \hat{r}_h^t(u, u') \,, \end{aligned}$$

and hence if $u' \in \mathcal{W}$ then

$$\hat{\mu}_h^{t+1}(u') = \frac{\mu_h^{t+1}(u')}{\pi_h^{t+1}} = \sum_{u \in \mathcal{W}} \hat{r}_h^t(u, u') + \frac{1}{\pi_h^{t+1}} \sum_{u \in \widetilde{\mathcal{W}}} \hat{r}_h^t(u, u') \,, \tag{5.36}$$

and if $u' \in \widetilde{\mathcal{W}}$ then

$$\hat{\mu}_h^{t+1}(u') = \mu_h^{t+1}(u') = \pi_h^{t+1} \sum_{u \in \mathcal{W}} \hat{r}_h^t(u, u') + \sum_{u \in \widetilde{\mathcal{W}}} \hat{r}_h^t(u, u') \,. \tag{5.37}$$

We will now show how we can update the quantities σ_h^t and $\tilde{\sigma}_h^t$ incrementally using the transition mass. Observe that for all $u \in \mathcal{Y}$,

$$\begin{aligned} \sum_{u' \in \mathcal{W}} \mu_h^t(u) r_t(u, u') &= \mu_h^t(u) \sum_{u' \in \mathcal{Y}} r_t(u, u') - \mu_h^t(u) \sum_{u' \in \widetilde{\mathcal{W}}} r_t(u, u') \\ &= \mu_h^t(u) - \mu_h^t(u) \sum_{u' \in \widetilde{\mathcal{W}}} r_t(u, u') \,, \end{aligned} \tag{5.38}$$

which when combined with (5.32) gives

$$\begin{aligned} \sigma_h^{t+1} &= \sum_{u' \in \mathcal{W}} \mu_h^{t+1}(u') \\ &= \psi_h^t \sum_{u \in \mathcal{W}} \sum_{u' \in \mathcal{W}} \mu_h^t(u) r_t(u, u') + \sum_{u \in \widetilde{\mathcal{W}}} \sum_{u' \in \mathcal{W}} \mu_h^t(u) r_t(u, u') \\ &= \psi_h^t \sum_{u \in \mathcal{W}} \left(\mu_h^t(u) - \mu_h^t(u) \sum_{u' \in \widetilde{\mathcal{W}}} r_t(u, u') \right) + \sum_{u \in \widetilde{\mathcal{W}}} \sum_{u' \in \mathcal{W}} \mu_h^t(u) r_t(u, u') \\ &= \psi_h^t \sigma_h^t - \psi_h^t \sum_{u \in \mathcal{W}} \sum_{u' \in \widetilde{\mathcal{W}}} \mu_h^t(u) r_t(u, u') + \sum_{u \in \widetilde{\mathcal{W}}} \sum_{u' \in \mathcal{W}} \mu_h^t(u) r_t(u, u') \\ &= \psi_h^t \sigma_h^t - \psi_h^t \sum_{u \in \mathcal{W}} \sum_{u' \in \widetilde{\mathcal{W}}} \pi_h^t \hat{\mu}_h^t(u) r_t(u, u') + \sum_{u \in \widetilde{\mathcal{W}}} \sum_{u' \in \mathcal{W}} \hat{\mu}_h^t(u) r_t(u, u') \\ &= \psi_h^t \sigma_h^t - \pi_h^{t+1} \sum_{u \in \mathcal{W}} \sum_{u' \in \widetilde{\mathcal{W}}} \hat{r}_h^t(u, u') + \sum_{u \in \widetilde{\mathcal{W}}} \sum_{u' \in \mathcal{W}} \hat{r}_h^t(u, u') \,. \end{aligned} \tag{5.39}$$

The same argument can be used to show that

$$\tilde{\sigma}_h^{t+1} = \tilde{\sigma}_h^t - \sum_{u \in \widetilde{\mathcal{W}}} \sum_{u' \in \mathcal{W}} \hat{r}_h^t(u, u') + \pi_h^{t+1} \sum_{u \in \mathcal{W}} \sum_{u' \in \widetilde{\mathcal{W}}} \hat{r}_h^t(u, u') \,. \tag{5.40}$$

We now define $\xi_h^t := \sum_{(u,u') \in \mathcal{W} \times \widetilde{\mathcal{W}}} \hat{r}_h^t(u, u')$ and $\tilde{\xi}_h^t := \sum_{(u,u') \in \widetilde{\mathcal{W}} \times \mathcal{W}} \hat{r}_h^t(u, u')$. Substituting these quantities into (5.39) and (5.40) then gives the following

updates:

$$\sigma_h^{t+1} = \psi_h^t \sigma_h^t - \pi_h^{t+1} \xi_h^t + \tilde{\xi}_h^t \,,$$

and

$$\tilde{\sigma}_h^{t+1} = \tilde{\sigma}_h^t - \tilde{\xi}_h^t + \pi_h^{t+1} \xi_h^t \,.$$

All that remains is the computation of ξ_h^{t+1} and $\tilde{\xi}_h^{t+1}$. For this, we introduce the sets $\mathcal{C}_t(h)$ and $\tilde{\mathcal{C}}_t(h)$, where

$$\mathcal{C}_t(h) := \{(u, u') \in \mathcal{W} \times \widetilde{\mathcal{W}} \,|\, \hat{r}_h^{t+1}(u, u') \neq \hat{r}_h^t(u, u')\} \,,$$

and

$$\tilde{\mathcal{C}}_t(h) := \{(u, u') \in \widetilde{\mathcal{W}} \times \mathcal{W} \,|\, \hat{r}_h^{t+1}(u, u') \neq \hat{r}_h^t(u, u')\} \,.$$

We will call these the "change sets." Now observe that

$$\begin{aligned} \xi_h^{t+1} &= \sum_{(u,u') \in \mathcal{W} \times \widetilde{\mathcal{W}}} \hat{r}_h^{t+1}(u, u') \\ &= \sum_{(u,u') \in \mathcal{W} \times \widetilde{\mathcal{W}}} \hat{r}_h^t(u, u') + \sum_{(u,u') \in \mathcal{C}_t(h)} \left(\hat{r}_h^{t+1}(u, u') - \hat{r}_h^t(u, u') \right) \\ &= \xi_h^t + \sum_{(u,u') \in \mathcal{C}_t(h)} \left(\hat{r}_h^{t+1}(u, u') - \hat{r}_h^t(u, u') \right) \,. \end{aligned} \tag{5.41}$$

The same argument can be used to show that

$$\tilde{\xi}_h^{t+1} = \tilde{\xi}_h^t + \sum_{(u,u') \in \tilde{\mathcal{C}}_t(h)} \left(\hat{r}_h^{t+1}(u, u') - \hat{r}_h^t(u, u') \right) \,. \tag{5.42}$$

This suggests that the key to designing fast methods lies in choosing a CGMC in which $\hat{r}_h^{t+1}(u, u') = \hat{r}_h^t(u, u')$ for almost all $(u, u') \in \mathcal{W} \times \widetilde{\mathcal{W}}$ and almost all $(u, u') \in \widetilde{\mathcal{W}} \times \mathcal{W}$. That is, the change sets $\mathcal{C}_t(h)$ and $\tilde{\mathcal{C}}_t(h)$ have low cardinality.

Given the ability to compute the change sets $\{\mathcal{C}_t(h), \tilde{\mathcal{C}}_t(h) \mid t \in [T], h \in \mathcal{H}\}$ and transition masses $\{\hat{r}_h^t \mid t \in [T], h \in \mathcal{H}\}$, or given an oracle which returns these quantities, our three routines can be expressed as shown in routines 7, 8,

Routine 7 initialize(h)

1: $\pi_h^1 \leftarrow 1$
2: $\sigma_h^1 \leftarrow \sum_{u \in \mathcal{W}} \iota(u)$
3: $\tilde{\sigma}_h^1 \leftarrow \sum_{u \in \widetilde{\mathcal{W}}} \iota(u)$
4: $\xi_h^1 \leftarrow \sum_{(u,u') \in \mathcal{W} \times \widetilde{\mathcal{W}}} \hat{r}_h^1(u,u')$
5: $\tilde{\xi}_h^1 \leftarrow \sum_{(u,u') \in \widetilde{\mathcal{W}} \times \mathcal{W}} \hat{r}_h^1(u,u')$

Routine 8 get(t,h)

1: **return** $(\sigma_h^t, \tilde{\sigma}_h^t)$

Routine 9 update(t,h)

1: $\pi_h^{t+1} \leftarrow \psi_h^t \pi_h^t$
2: $\sigma_h^{t+1} \leftarrow \psi_h^t \sigma_h^t - \pi_h^{t+1} \xi_h^t + \tilde{\xi}_h^t$
3: $\tilde{\sigma}_h^{t+1} \leftarrow \tilde{\sigma}_h^t - \tilde{\xi}_h^t + \pi_h^{t+1} \xi_h^t$
4: $\xi_h^{t+1} \leftarrow \xi_h^t + \sum_{(u,u') \in \mathcal{C}_t(h)} (\hat{r}_h^{t+1}(u,u') - \hat{r}_h^t(u,u'))$
5: $\tilde{\xi}_h^{t+1} \leftarrow \tilde{\xi}_h^t + \sum_{(u,u') \in \tilde{\mathcal{C}}_t(h)} (\hat{r}_h^{t+1}(u,u') - \hat{r}_h^t(u,u'))$

and 9.

If we assume the existence of such an oracle, then note that this now requires a time of $\mathcal{O}(|\mathcal{C}_t(h)| + |\tilde{\mathcal{C}}_t(h)|)$ per trial (per policy). In the next section, we will formally define our CGMC and show how to compute these quantities, removing the need for such an oracle.

5.6 AdaptLTM - A CGMC for Adaptive LTM

In this section we introduce our chosen CGMC for the problem of adaptive LTM, and the resulting algorithm, which we call AdaptLTM. This CGMC can be seen as a modified version of $\mathcal{M}_A$, described in Section 5.5.2. Recall that this algorithm will correspond to Algorithm 7 with the three routines **initialize**, **get**, and **update** defined appropriately.

We describe the graphical structure of our CGMC using the function $\mathcal{A} : \mathcal{Y} \to 2^{\mathcal{Y}}$ defined by

$$\mathcal{A}(u) := \{u' \in \mathcal{Y} \mid \exists t \in \mathbb{N} : r_t(u,u') \neq 0\},$$

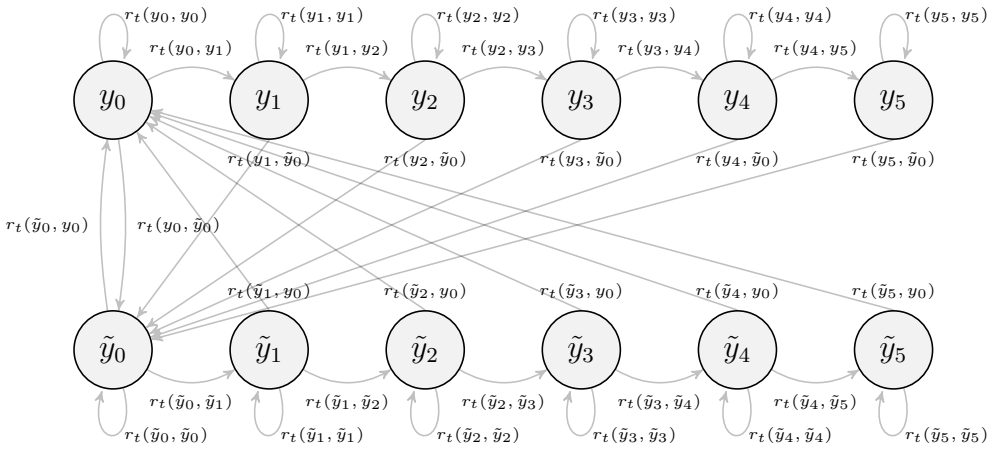


Figure 5.5: A finite example of our circadian-generating Markov chain for $q \in \{0, \ldots, 5\}$. For each state y_q we have $\mathcal{A}(y_q) = \{y_q, y_{q+1}, \tilde{y}_0\}$, and similarly for each state $\tilde{y}_q$ we have $\mathcal{A}(\tilde{y}_q) := \{\tilde{y}_q, \tilde{y}_{q+1}, y_0\}$, where $\mathcal{A}(u) := \{u' \in \mathcal{Y} \mid \exists t \in \mathbb{N} : r_t(u, u') \neq 0\}$.

for all $u \in \mathcal{Y}$. Just as in $\mathcal{M}_A$, we denote our set of awake states, $\mathcal{W}$, by $\mathcal{W} := \{y_q \mid q \in \mathbb{N} \cup \{0\}\}$ and our set of asleep states, $\widetilde{\mathcal{W}}$, by $\widetilde{\mathcal{W}} := \{\tilde{y}_q \mid q \in \mathbb{N} \cup \{0\}\}$. We can thus interpret the different states $y_0, y_1, \ldots$ in $\mathcal{W}$ as different levels of "wakefulness," where state y_{q+1} is interpreted as being more *wide awake* than state y_q. Similarly state $\tilde{y}_{q+1}$ is interpreted as being more *deeply asleep* than state $\tilde{y}_q$. Intuitively, being more awake (or more asleep) will mean that it is *harder* (less likely) to fall asleep (or wake up).

The graphical structure of our CGMC is given by $\mathcal{A}(y_q) := \{y_q, y_{q+1}, \tilde{y}_0\}$, and $\mathcal{A}(\tilde{y}_q) := \{\tilde{y}_q, \tilde{y}_{q+1}, y_0\}$, for all $q \in \mathbb{N} \cup \{0\}$. Note that our graphical structure is *sparse*; if the Markov chain is currently in an awake state y_q it can either:

1. Stay in the current level of wakefulness (remain in state y_q).

2. Become more wide awake (transition to state y_{q+1}).

3. Fall asleep (transition to state $\tilde{y}_0$).

The converse is true for being in state $\tilde{y}_q$ with a reversal of the notions of asleep and awake (see Figure 5.5 for an illustration). The sparsity of our CGMC will be crucial to the change sets $\mathcal{C}_t(h)$ and $\tilde{\mathcal{C}}_t(h)$ being of low cardinality and our

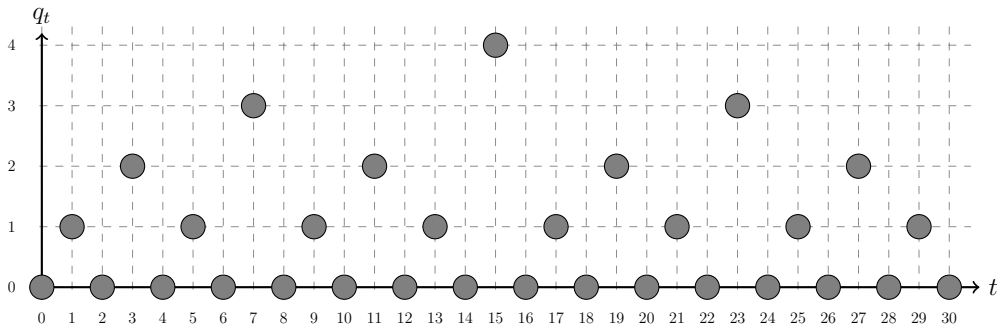


Figure 5.6: A plot of the tree-height function for $t = 0, \ldots, 30$.

algorithm being computationally efficient.

5.6.1 The Transition Function

We now define our transition function, $\boldsymbol{r}$, for our CGMC. The transition function is chosen to achieve a good regret bound, and has a complex form to ensure that the change sets $\mathcal{C}_t(h)$ and $\tilde{\mathcal{C}}_t(h)$ have low cardinality. We show how this is done in Section 5.6.2.

For our transition function, we will require a parameter $\alpha \in (0, 1)$. Note that apart from the learning rate, η, this is the only parameter that our algorithm will require. Furthermore, we suggest a tuning of $\alpha = 1/4$, which gives a good regret bound and performs reasonably well in our experiments.

For all $t \in \mathbb{N} \cup \{0\}$ we define the tree-height function as

$$\text{tree-height}(t) := \max\{q \in \mathbb{N} \cup \{0\} \mid \exists m \in \mathbb{N} : t + 1 = m2^q\},$$

and for simplicity we denote $q_t := \text{tree-height}(t)$. The quantity q_t is the largest integer q such that $t + 1$ is a multiple of 2^q. See Figure 5.6 for a visual representation of this function. Intuitively the tree-height function gives the height of a node of a binary tree "across time," where tree-height(t) is the height of the t^{th} node when counting nodes "from left to right." We also define for all $q \in \mathbb{N}$,

$$k_{t,q} := t - \max\{s \in [t] \mid q_{s-1} = q - 1\}.$$

Note that every trial $t \in \mathbb{N}$ will have a corresponding "tree height" (q_t) which

will be used by our algorithm. Thus intuitively, for a given "level" q, on a given trial t, $k_{t,q}$ is the number of trials since the last time that the "tree height" was $q-1$ (see Figure 5.7).

We finally combine these quantities to define

$$\omega_{t,q} := \frac{\alpha}{2^q - \alpha k_{t,q}}\,, \tag{5.43}$$

which we will discuss later.

Our transition function is then defined as follows. For $y \in \mathcal{Y}$ our initial state function is defined as

$$\iota(y) := \begin{cases} \alpha & y = y_0\,, \\ 1-\alpha & y = \tilde{y}_1\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

Our transition function is then defined such that for all $t \in \mathbb{N}$ we have:

- $r_t(y_0, \tilde{y}_0) := r_t(\tilde{y}_0, y_0) := \alpha$.
- $r_t(y_0, y_0) := r_t(\tilde{y}_0, \tilde{y}_0) := [\![q_t \neq 0]\!](1-\alpha)$.
- $r_t(y_0, y_1) := r_t(\tilde{y}_0, \tilde{y}_1) := [\![q_t = 0]\!](1-\alpha)$.

Additionally for all $q \in \mathbb{N}$ we have:

- $r_t(y_q, \tilde{y}_0) := r_t(\tilde{y}_q, y_0) := \omega_{t,q}$.
- $r_t(y_q, y_q) := r_t(\tilde{y}_q, \tilde{y}_q) := [\![q_t \neq q]\!](1-\omega_{t,q})$.
- $r_t(y_q, y_{q+1}) := r_t(\tilde{y}_q, \tilde{y}_{q+1}) := [\![q_t = q]\!](1-\omega_{t,q})$.

This transition function has the following properties. We describe these properties by considering a particle's trajectory through the Markov chain. We refer to the *depth* of a given state given by the subscript q, where the state y_{q+1} is considered deeper than y_q and so on (see Figure 5.5).

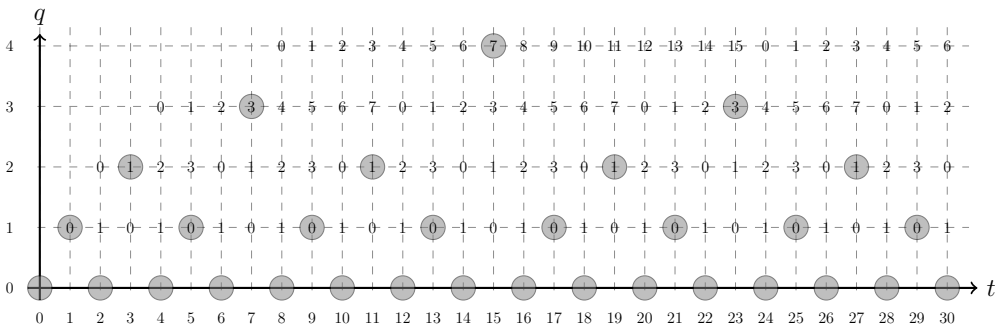
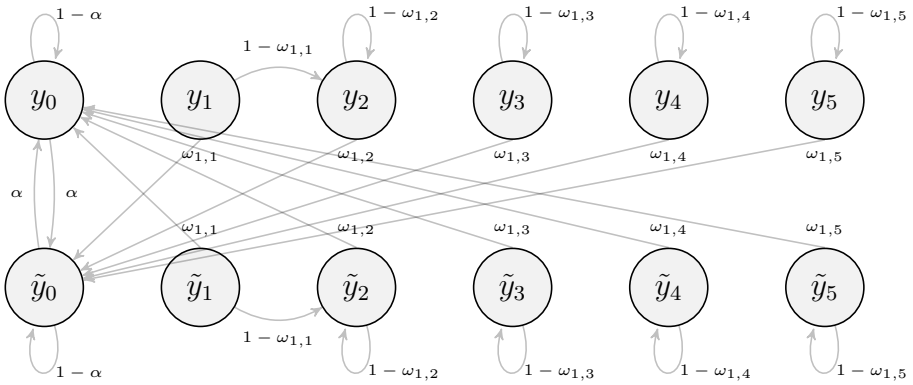
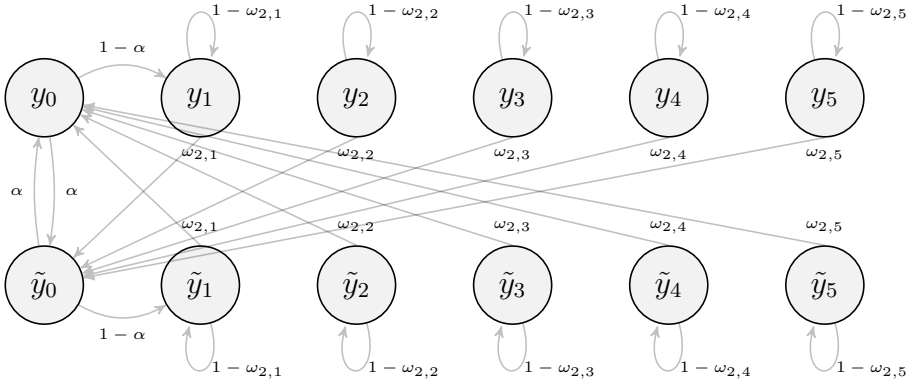


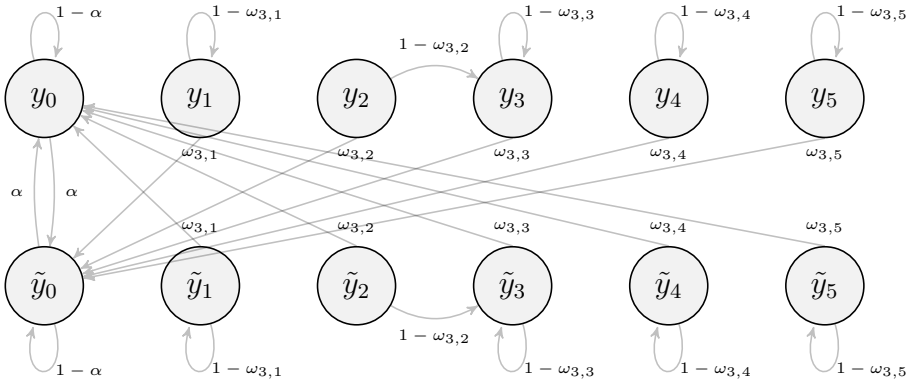
Figure 5.7: At coordinates (t, q) the value of $k_{t,q}$ is shown. The tree-height function is additionally shown (gray circles).



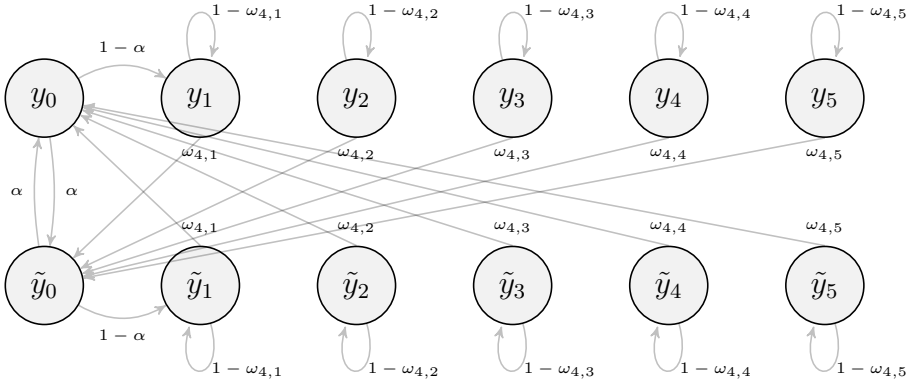
(a) $t = 1, q_t = 1$.



(b) $t = 2, q_t = 0$.



(c) $t = 3, q_t = 2$.



(d) $t = 4, q_t = 0$.

Figure 5.8: Graphical representations of our transition function for $t = 1, \ldots, 4$ and the corresponding tree-height q_t on each trial.

Firstly, as discussed, observe that the particle can only "fall asleep" (that is, transition from a state $y \in \mathcal{W}$ to a state $\tilde{y} \in \widetilde{\mathcal{W}}$) by transitioning to $\tilde{y}_0$. The converse is true for "waking up" which is only possible by transitioning from a state $\tilde{y} \in \widetilde{\mathcal{W}}$ to y_0. Secondly, consider a particle's trajectory as it remains awake (or equivalently stays asleep). Assume therefore that a particle is in state y_0, and it is to reach state y_{q^*} for some $q^* > 1$ without falling asleep (transitioning to $\tilde{y}_0$). Observe that $r_t(y_q, y_{q+1}) \neq 0$ only if $q_t = q$. Thus the particle may only transition from y_q to a deeper state y_{q+1} when $q_t = q$. Furthermore, observe that if $q_t = q$ then $r_t(y_q, y_q) = 0$, and the particle *must* transition deeper (given that it does not fall asleep). The tree-height, q_t, therefore determines on trial t the depth at which a transition to a deeper state can occur. The binary tree-like structure of the tree-height function means that the particle must spend either 2^{q-1} trials or $2^q + 2^{q-1}$ trials in the state y_q before transitioning to state y_{q+1}. It therefore takes an exponential amount of time for the particle to reach state y_{q^*}. See Figure 5.8 for an illustration of this behavior.

We now turn our attention to the quantity $\omega_{t,q}$ used to define our transition probabilities. Recall that our goal in designing a computationally efficient CGMC is to ensure that the change sets $\mathcal{C}_t(h)$ and $\tilde{\mathcal{C}}_t(h)$ have low cardinality. That is, ensuring that $\hat{r}_h^{t+1}(u, u') = \hat{r}_h^t(u, u')$ for almost all $(u, u') \in \mathcal{W} \times \widetilde{\mathcal{W}}$ and almost all $(u, u') \in \widetilde{\mathcal{W}} \times \mathcal{W}$. Given the graphical structure of our CGMC, this means that we require $\hat{r}_h^{t+1}(u, \tilde{y}_0) = \hat{r}_h^t(u, \tilde{y}_0)$ for almost all $u \in \mathcal{W}$ and $\hat{r}_h^{t+1}(\tilde{u}, y_0) = \hat{r}_h^t(\tilde{u}, y_0)$ for almost all $\tilde{u} \in \widetilde{\mathcal{W}}$ (see Figure 5.5).

Recall that each trial t has a corresponding tree-height, q_t. For a given level $q \in \mathbb{N}$ in our CGMC, there will be three events we must consider: when $q = q_t$, when $q = q_t + 1$, and when $q \notin \{q_t, q_t + 1\}$. Note that when $q = q_t$ we have $\hat{r}_h^t(y_q, y_q) = 0$, and when $q = q_t + 1$ we have $\hat{r}_h^t(y_{q-1}, y_q) \neq 0$. In the next section we will see that it is on these trials (i.e., when $q \in \{q_t, q_t + 1\}$) that we have $(y_q, \tilde{y}_0) \in \mathcal{C}_t(h)$ and $(\tilde{y}_q, y_0) \in \tilde{\mathcal{C}}_t(h)$. We first consider the third case ($q \notin \{q_t, q_t + 1\}$). Observe that for a given level q, such trials occur in contiguous segments. We will now show that for segments of trials $[s, s']$ where

$q \notin \{q_t, q_t + 1\}$ for $t \in [s, s']$ that $(y_q, \tilde{y}_0) \notin \mathcal{C}_t(h)$ and $(\tilde{y}_q, y_0) \notin \tilde{\mathcal{C}}_t(h)$.

Let $[s, s']$ be such a segment. Note that for all $q \in \mathbb{N}$ we have $r_t(y_q, \tilde{y}_0) = \omega_{t,q}$, and thus our transition mass is given by

$$\begin{aligned} \hat{r}_h^t(y_q, \tilde{y}_0) &= \hat{\mu}_h^t(y_q) r_t(y_q, \tilde{y}_0) \\ &= \hat{\mu}_h^t(y_q)\omega_{t,q}\,, \end{aligned}$$

and note that for all $q \in \mathbb{N}$ such that $q \neq q_t$ we have $r_t(y_q, y_q) = 1 - \omega_{t,q}$, which implies that for $t \in [s, s']$ we have

$$\begin{aligned} \hat{\mu}_h^{t+1}(y_q) &= \frac{\mu_h^{t+1}(y_q)}{\pi_h^{t+1}} \\ &= \frac{\psi_h^t \mu_h^t(y_q) r_t(y_q, y_q)}{\pi_h^{t+1}} \\ &= \frac{\psi_h^t \pi_h^t \hat{\mu}_h^t(y_q) r_t(y_q, y_q)}{\pi_h^{t+1}} \\ &= \hat{\mu}_h^t(y_q) r_t(y_q, y_q) \\ &= \hat{\mu}_h^t(y_q)(1 - \omega_{t,q})\,. \end{aligned}$$

Thus $\hat{r}_h^t(y_q, \tilde{y}_0) = \hat{r}_h^{t+1}(y_q, \tilde{y}_0)$ if $\hat{\mu}_h^t(y_q)\omega_{t,q} = \hat{\mu}_h^{t+1}(y_q)\omega_{t+1,q}$, which means that we require

$$\hat{\mu}_h^t(y_q)\omega_{t,q} = \hat{\mu}_h^t(y_q)(1 - \omega_{t,q})\omega_{t+1,q}\,.$$

In other words, our transition mass from state y_q to state $\tilde{y}_0$ on trial t is a proportion $(\omega_{t,q})$ of our original "mass" $\hat{\mu}_h^t(y_q)$. On the next trial, our transition mass from y_q to $\tilde{y}_0$ is then a proportion $(\omega_{t+1,q})$ of the remaining mass $\hat{\mu}_h^t(y_q)(1 - \omega_{t,q})$. Thus for these quantities to be equal during these segments of trials, we require that on each trial t, the mass $\hat{\mu}_h^t(y_q)$ decrease by a multiplicative factor, $\omega_{t,q}$, such that the *amount* transitioned is constant on each trial t. The quantity $\omega_{t,q}$ is designed to achieve such behavior. As an illustration, consider a unit of mass that is reduced by 1/4 to 0.75, then reduced by 1/3 to 0.5, then finally reduced by 1/2 to 0.25. On each step we lose 0.25 of mass. This is exactly the principle behind the choice of $\omega_{t,q}$. Indeed, with $\alpha = 1$

and $q = 2$ for example, on trials $t = 6, 7, 8$ we have $k_{t,q} = 0, 1, 2$ respectively and we recover exactly these factors ($\omega_{t,q} = 1/2^q$, $1/(2^q - 1)$, $1/(2^q - 2)$). Note however that $\alpha \in \{0, 1\}$ is not a valid choice for our algorithm given the initial state function $\iota(y)$.[2]

The form of $\omega_{t,q}$ also means that for any $t, q \in \mathbb{N}$, $\omega_{t,q} \leq \omega_{t,q+1}$. This means that the deeper the particle is in our CGMC, the *harder* it is to fall asleep (or wake up). Furthermore, the effect of decreasing α is that the quantity $\omega_{t,q}$ of each level q (and thus the transition mass) decreases, further making it harder to fall asleep (or wake up) for more wide awake (or deeply asleep) states. Intuitively, the graphical structure of our CGMC and the functional form of $\omega_{t,q}$ is how our inductive bias of favoring circadian patterns consisting of periods of sleeping/waking frequently (and long periods of deep sleep) is introduced in our prior.

In the next section, we further consider the mechanics behind $\omega_{t,q}$ and how we exploit this to obtain an efficient implementation of our algorithm AdaptLTM.

5.6.2 Implementing AdaptLTM

Recall that in Section 5.4.3 we assumed the existence of an oracle to give the change sets $\{\mathcal{C}_t(h), \tilde{\mathcal{C}}_t(h) \mid t \in [T], h \in \mathcal{H}\}$ and the transition masses $\{\hat{r}_h^t \mid t \in [T], h \in \mathcal{H}\}$. Now that our CGMC has been defined in Section 5.6.1, our goal is to derive the computation of these quantities explicitly, and thus re-write routines 7, 8, and 9 for AdaptLTM.

For all $t, q \in \mathbb{N}$ we first define

$$l(t, q) := \max \{s \in [t] \mid q_{s-1} \in \{q - 1, q\}\}.$$

[2]We point out, however, that with an initial state function of, for example, $\iota(y_0) = \iota(\tilde{y}_0) = 1/2$, and $\iota(y_q) = \iota(\tilde{y}_q) = 0$ for $q \in \mathbb{N}$, then $\alpha = 1$ is a valid choice. Furthermore, note that this would put a non-zero prior probability on "oscillating" circadian patterns $\boldsymbol{\gamma} = (0, 1, 0, 1, \ldots)$, and $\boldsymbol{\gamma} = (1, 0, 1, 0, \ldots)$ only, and this corresponds exactly with the "2-expert constant-switching" learning problem we considered briefly at the end of Chapter 4 (that is, it is identical to Share-θ with $\alpha = \theta = 1$). Interestingly, this is achieved in our algorithm with a single parameter. Our given choice of the initial state function for AdaptLTM was made for convenience in proving the regret bound.

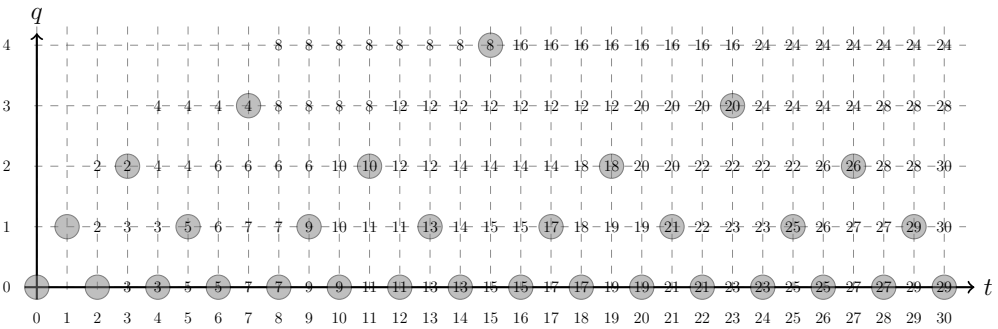


Figure 5.9: At coordinates (t, q) the value of $l(t, q)$ is shown. The tree-height function is additionally shown (gray circles).

Intuitively for a given depth, q, in our Markov chain, on trial t the quantity $l(t, q)$ is the most recent trial before which either a "deepening" transition from y_q to y_{q+1} could have occurred, or a deepening transition from y_{q-1} to y_q could have occured (see Figure 5.9).

For $h \in \mathcal{H}$ we then define the quantities:

$$\beta_{q,h}^t := \frac{\hat{\mu}_h^{l(t,q)}(y_q)}{2^q}$$

and

$$\tilde{\beta}_{q,h}^t := \frac{\hat{\mu}_h^{l(t,q)}(\tilde{y}_q)}{2^q}\,.$$

We also define $\beta_{0,h}^t := \hat{\mu}_h^t(y_0)$ and $\tilde{\beta}_{0,h}^t = \hat{\mu}_h^t(\tilde{y}_0)$. We now fix some arbitrary $q \in \mathbb{N}$ and show how we compute the transition mass $\hat{r}_h^t(y_q, u)$ for $t \in \mathbb{N}$ and $u \in \mathcal{Y}$ for a given policy $h \in \mathcal{H}$. First, note that they are only ever non-zero for $u \in \mathcal{A}(y_q) = \{y_q, y_{q+1}, \tilde{y}_0\}$. From (5.36) and the fact that $y_q \in \mathcal{A}(u)$ if and only if $u \in \{y_{q-1}, y_q\}$, we have

$$\begin{aligned}
\hat{\mu}_h^{t+1}(y_q) &= \sum_{u \in \mathcal{W}} \hat{r}_h^t(u, y_q) \\
&= \hat{r}_h^t(y_{q-1}, y_q) + \hat{r}_h^t(y_q, y_q) \\
&= r_t(y_{q-1}, y_q)\hat{\mu}_h^t(y_{q-1}) + r_t(y_q, y_q)\hat{\mu}_h^t(y_q)\,. \qquad (5.44)
\end{aligned}$$

We now require the following Proposition, which we will use in deriving

the update of our algorithm.

Proposition 28. *Given some fixed value $q \in \mathbb{N}$ and any arbitrary $t \in \mathbb{N}$ such that $q_t = q - 1$, then for all $s \in [2^q]$,*

$$\hat{\mu}_h^{t+s}(y_q) = (2^q - (s-1)\alpha)\beta_{q,h}^{t+s}\,, \tag{5.45}$$

for all $h \in \mathcal{H}$.

Proof. Given some arbitrary $t \in \mathbb{N}$ such that $q_t = q - 1$, note that we have two cases: either $q_{(t+2^{q-1})} \neq q$, or $q_{(t+2^{q-1})} = q$. We consider these two cases separately.

Case 1: $(q_{(t+2^{q-1})} \neq q)$. In this case, we use a proof by induction, taking the inductive hypothesis that (5.45) holds for all $s \in [2^q]$. Note that this is true for $s = 1$ since $l(t+1, q) = t + 1$, which implies that $\beta_{q,h}^{t+1} = \hat{\mu}_h^{t+1}(y_q)/2^q$. Now suppose it holds for some $s \in [2^q - 1]$. Note that since $q_{(t+2^{q-1})} \neq q$ then $q_{t+s} \notin \{q-1, q\}$ which implies that $r_{t+s}(y_{q-1}, y_q) = 0$ and $r_{t+s}(y_q, y_q) = 1 - \omega_{(t+s),q}$. Substituting this into (5.44) and substituting that into our inductive hypothesis gives

$$\begin{aligned}
\hat{\mu}_h^{t+s+1}(y_q) &= r_{t+s}(y_q, y_q)\hat{\mu}_h^{t+s}(y_q) \\
&= (1 - \omega_{t+s,q})\hat{\mu}_h^{t+s}(y_q) \\
&= (1 - \omega_{t+s,q})(2^q - (s-1)\alpha)\beta_{q,h}^{t+s} \\
&= \left(1 - \frac{\alpha}{2^q - \alpha k_{t+s,q}}\right)(2^q - (s-1)\alpha)\beta_{q,h}^{t+s} \\
&= \left(1 - \frac{\alpha}{2^q - (s-1)\alpha}\right)(2^q - (s-1)\alpha)\beta_{q,h}^{t+s} \\
&= (2^q - s\alpha)\beta_{q,h}^{t+s},.
\end{aligned}$$

Where we have used the fact that $k_{t+s,q} = (t+s) - (t+1) = s - 1$ for $s \in [2^q]$ when $q_t = q - 1$. Now by noting that $l(t+s+1, q) = t + 1 = l(t+s, q)$ for $s \in [2^q]$ then this implies that $\beta_{q,h}^{t+s+1} = \beta_{q,h}^{t+s}$, and we have therefore shown that the inductive hypothesis holds for $s + 1$, and hence holds for all $s \in [2^q]$. This

proves that (5.45) holds for case 1.

Case 2: $(q_{(t+2^{q-1})} = q)$. Firstly, in the case that $q_{(t+2^{q-1})} = q$, the same argument as Case 1 can be used to show that (5.45) holds for $s \in [2^{q-1}]$. However, since $q_{(t+2^{q-1})} = q$ then this implies that $\hat{r}_h^{t+2^{q-1}}(y_{q-1}, y_q) = 0$ and $\hat{r}_h^{t+2^{q-1}}(y_q, y_q) = 0$. From (5.44) this implies that

$$\hat{\mu}_h^{t+2^{q-1}+1}(y_q) = 0\,. \tag{5.46}$$

Additionally note that for all $s' \in [2^{q-1}+1, 2^q - 1]$ we have $q_{t+s'} \neq q-1$ which implies that $\hat{r}_h^{t+s'}(y_{q-1}, y_q) = 0$, and hence, using (5.46) and (5.44), a simple inductive argument on s gives

$$\hat{\mu}_h^{t+s}(y_q) = 0 \tag{5.47}$$

for all $s \in [2^{q-1}+1, 2^q]$. For all such s we have $l(t+s, q) = t + 2^{q-1} + 1$, so by (5.46) we have

$$\beta_{q,h}^{t+s} = \frac{\hat{\mu}_h^{t+2^{q-1}+1}(y_q)}{2^q} = 0\,,$$

which using (5.47) gives $\hat{\mu}_h^{t+s}(y_q) = 0 = (2^q - (s-1)\alpha)\beta_{q,h}^{t+s}$ as desired. □

Using Proposition 28 and the definition of $\hat{r}^t$, we therefore have for all $t \in \mathbb{N}$ with $q_t = q - 1$, for all $s \in [2^q]$ that

$$\begin{aligned}
\hat{r}_h^{t+s}(y_q, \tilde{y}_0) &= r_{t+s}(y_q, \tilde{y}_0)\hat{\mu}_h^{t+s}(y_q) \\
&= \omega_{t,q}(2^q - (s-1)\alpha)\beta_{q,h}^{t+s} \\
&= \frac{\alpha}{2^q - \alpha k_{t+s,q}}(2^q - (s-1)\alpha)\beta_{q,h}^{t+s} \\
&= \frac{\alpha}{2^q - \alpha(s-1)}(2^q - (s-1)\alpha)\beta_{q,h}^{t+s} \\
&= \alpha\beta_{q,h}^{t+s}\,,
\end{aligned}$$

where we have again used the fact that $k_{t+s,q} = (t+s) - (t+1) = s - 1$. The same argument for state $\tilde{y}_q$ also gives $\hat{r}_h^{t+s}(\tilde{y}_q, y_0) = \alpha\tilde{\beta}_{q,h}^{t+s}$. We therefore have,

for all $t \in \mathbb{N}$ that

$$\hat{r}_h^t(y_q, \tilde{y}_0) = \alpha \beta_{q,h}^t \,, \tag{5.48}$$

and

$$\hat{r}_h^t(\tilde{y}_q, y_0) = \alpha \tilde{\beta}_{q,h}^t \,. \tag{5.49}$$

Now note that since $\beta_{q,h}^{t'+1} = \beta_{q,h}^{t'}$ for all $t' \in \mathbb{N}$ with $q_{t'} \notin \{q-1, q\}$, this implies that for all $t \in \mathbb{N}$ we have $(y_q, \tilde{y}_0) \in \mathcal{C}_t(h)$ if and only if $q_t \in \{q-1, q\}$. Similarly, we have $(\tilde{y}_q, y_0) \in \tilde{\mathcal{C}}_t(h)$ if and only if $q_t \in \{q-1, q\}$. Recall that our CGMC is defined such that falling asleep (or waking up) is only possible by transitioning to $\tilde{y}_0$ (or y_0, respectively). That is, for all $t', q' \in \mathbb{N}$ we have $\hat{r}_h^{t'}(y_q, \tilde{y}_{q'}) = 0$ and hence $(y_q, \tilde{y}_{q'}) \notin \mathcal{C}_t(h)$, and similarly $(\tilde{y}_q, y_{q'}) \notin \tilde{\mathcal{C}}_t(h)$. We conclude that

$$\mathcal{C}_t(h) = \{(y_0, \tilde{y}_0), (y_{q_t}, \tilde{y}_0), (y_{q_t+1}, \tilde{y}_0)\} \,, \tag{5.50}$$

and

$$\tilde{\mathcal{C}}_t(h) = \{(\tilde{y}_0, y_0), (\tilde{y}_{q_t}, y_0), (\tilde{y}_{q_t+1}, y_0)\} \,. \tag{5.51}$$

We can now finally express the updates (5.41) and (5.42) in terms of the quantities $\{\beta_{q,h}^t, \tilde{\beta}_{q,h}^t \mid t \in \mathbb{N}, q \in \mathbb{N} \cup \{0\}\}$ using (5.48), (5.49), (5.50), and (5.51). Thus for all $t \in \mathbb{N}$ we have

$$\begin{aligned} \xi_h^{t+1} &= \xi_h^t + \sum_{(u,u') \in \mathcal{C}_t(h)} \left(\hat{r}_h^{t+1}(u, u') - \hat{r}_h^t(u, u') \right) \\ &= \xi_h^t + \sum_{q \in \{0, q_t, q_{t+1}\}} \alpha \left(\beta_{q,h}^{t+1} - \beta_{q,h}^t \right) \,, \end{aligned} \tag{5.52}$$

and

$$\begin{aligned} \tilde{\xi}_h^{t+1} &= \tilde{\xi}_h^t + \sum_{(u,u') \in \tilde{\mathcal{C}}_t(h)} \left(\hat{r}_h^{t+1}(u, u') - \hat{r}_h^t(u, u') \right) \\ &= \tilde{\xi}_h^t + \sum_{q \in \{0, q_t, q_{t+1}\}} \alpha \left(\tilde{\beta}_{q,h}^{t+1} - \tilde{\beta}_{q,h}^t \right) \,, \end{aligned} \tag{5.53}$$

where duplicate elements in the set $\{0, q_t, q_{t+1}\}$ are only counted once.

5.6.2.1 Updating $\beta^t_{q,h}$ and $\tilde{\beta}^t_{q,h}$

All that remains in our implementation is the update of the variables $\beta^t_{q,h}$ and $\tilde{\beta}^t_{q,h}$ for all $t \in \mathbb{N}$ and $q \in \mathbb{N} \cup \{0\}$. We first fix some $t, q \in \mathbb{N}$ (we address the case that $q = 0$ later). Recall that if $q_t \notin \{q-1, q\}$ we simply have $\beta^{t+1}_{q,h} = \beta^t_{q,h}$. We therefore consider the cases where $q_t \in \{q-1, q\}$, noting that in both of these cases we have $l(t+1, q) = t+1$.

Case 1: $(q_t = q)$. In the case that $q_t = q$ we have $\hat{r}^t_h(y_{q-1}, y_q) = 0$, and $\hat{r}^t_h(y_q, y_q) = 0$, and thus straightforwardly we have

$$\beta^{t+1}_{q,h} = \frac{\hat{\mu}^{l(t+1,q)}_h(y_q)}{2^q} = \frac{\hat{\mu}^{t+1}_h(y_q)}{2^q} = 0\,.$$

Case 2: $(q_t = q-1)$. The case that $q_t = q-1$ is slightly more complicated. First, note that

$$\begin{aligned}\hat{r}^t_h(y_{q-1}, y_q) &= \hat{\mu}^t_h(y_{q-1}) r_t(y_{q-1}, y_q) \\ &= [\![q=1]\!](1-\alpha)\hat{\mu}^t_h(y_{q-1}) + [\![q>1]\!](1-\omega_{t,q-1})\hat{\mu}^t_h(y_{q-1})\,. \quad (5.54)\end{aligned}$$

This gives us two further cases to consider: when $q = 1$ and when $q > 1$. In the case that $q = 1$ we have $\hat{\mu}^t_h(y_{q-1}) = \hat{\mu}^t_h(y_0) = \beta^t_{0,h}$, and so by (5.54) we have

$$\hat{r}^t_h(y_{q-1}, y_q) = \hat{r}^t_h(y_0, y_1) = (1-\alpha)\beta^t_{0,h}\,. \quad (5.55)$$

In the case that $q > 1$, note that since $q_t = q-1$ we have that $t+1$ is a multiple of 2^{q-1} so $t+1-2^{q-2}$ is a multiple of 2^{q-2}, but not 2^{q-1}. In other words we have $t = s' + 2^{q-2}$ for some s' with $q_{s'} = q-2$. Fix such an s', from Proposition 28 we have for all $s \in [2^{q-1}]$ that $\hat{\mu}^{s'+s}_h(y_{q-1}) = (2^{q-1} - (s-1)\alpha)\beta^{s'+s}_{q-1,h}$ and thus by plugging in $s = 2^{q-2}$ we have

$$\hat{\mu}^t_h(y_{q-1}) = \left(2^{q-1} - (2^{q-2}-1)\alpha\right)\beta^t_{q-1,h}\,.$$

Substituting this into (5.54), and noting that

$$\begin{aligned} k_{t,q-1} &= t - \max\{s \in [t] \mid q_{s-1} = q-2\} \\ &= t - (s'+1) \\ &= 2^{q-2} - 1 \end{aligned}$$

then gives

$$\begin{aligned} \hat{r}_h^t(y_{q-1}, y_q) &= (1-\omega_{t,q-1})(2^{q-1} - (2^{q-2}-1)\alpha)\beta_{q-1,h}^t \\ &= \left(1 - \frac{\alpha}{2^{q-1} - \alpha k_{t,q-1}}\right)(2^{q-1} - (2^{q-2}-1)\alpha)\beta_{q-1,h}^t \\ &= \left(1 - \frac{\alpha}{2^{q-1} - (2^{q-2}-1)\alpha}\right)(2^{q-1} - (2^{q-2}-1)\alpha)\beta_{q-1,h}^t \\ &= \left(\frac{2^{q-1} - \alpha 2^{q-2}}{2^{q-1} - (2^{q-2}-1)\alpha}\right)(2^{q-1} - (2^{q-2}-1)\alpha)\beta_{q-1,h}^t \\ &= 2^{q-1}\left(1 - \frac{\alpha}{2}\right)\beta_{q-1,h}^t . \end{aligned} \tag{5.56}$$

Now that we have shown how to compute $\hat{r}_h^t(y_{q-1}, y_q)$ in terms of $\beta_{q-1,h}^t$, we turn our attention to computing $\hat{r}_h^t(y_q, y_q)$ in terms of $\beta_{q,h}^t$. Recall that since $q_t = q-1$, then t is equal to $s + 2^q$ for some $s < t$ such that $q_s = q-1$. Additionally, we have $k_{t,q} = 2^q - 1$. From Proposition 28 we then have $\hat{\mu}_h^t(y_q) = (2^q - (2^q-1)\alpha)\beta_{q,h}^t$ and therefore

$$\begin{aligned} \hat{r}_h^t(y_q, y_q) &= r_t(y_q, y_q)\hat{\mu}_h^t(y_q) \\ &= (1-\omega_{t,q})\hat{\mu}_h^t(y_q) \\ &= (1-\omega_{t,q})(2^q - (2^q-1)\alpha)\beta_{q,h}^t \\ &= \left(1 - \frac{\alpha}{2^q - \alpha k_{t,q}}\right)(2^q - (2^q-1)\alpha)\beta_{q,h}^t \\ &= \left(1 - \frac{\alpha}{2^q - (2^q-1)\alpha}\right)(2^q - (2^q-1)\alpha)\beta_{q,h}^t \\ &= \left(\frac{(1-\alpha)2^q}{2^q - (2^q-1)\alpha}\right)(2^q - (2^q-1)\alpha)\beta_{q,h}^t \\ &= 2^q(1-\alpha)\beta_{q,h}^t . \end{aligned} \tag{5.57}$$

Having expressed $\hat{r}_h^t(y_{q-1}, y_q)$ and $\hat{r}_h^t(y_q, y_q)$ in terms of $\beta_{q-1,h}^t$ and $\beta_{q,h}^t$ respectively, we now combine (5.44), (5.56), and (5.57) to give

$$\begin{aligned}
\beta_{q,h}^{t+1} &= \frac{\hat{\mu}_h^{l(t+1,q)}(y_q)}{2^q} \\
&= \frac{\hat{\mu}_h^t(y_q)}{2^q} \\
&= \frac{\hat{r}_h^t(y_{q-1}, y_q)}{2^q} + \frac{\hat{r}_h^t(y_q, y_q)}{2^q} \\
&= [\![q = 1]\!] \left(\frac{1-\alpha}{2}\right) \beta_{q-1,h}^t + [\![q > 1]\!] \left(\frac{2-\alpha}{4}\right) \beta_{q-1,h}^t + (1-\alpha)\beta_{q,h}^t \,.
\end{aligned}$$

In summary, our update for $\beta_{q,h}^{t+1}$ and $\tilde{\beta}_{q,h}^{t+1}$ for $q \in \mathbb{N} \cup \{0\}$ is as follows. For $q \notin \{0, q_t, q_t + 1\}$ we have $\beta_{q,h}^{t+1} = \beta_{q,h}^t$ and $\tilde{\beta}_{q,h}^{t+1} = \tilde{\beta}_{q,h}^t$. For $q \in \{0, q_t, q_t + 1\}$ we have the following. When $q = q_t + 1$, we have on trial t

$$\beta_{q_t+1,h}^{t+1} = \begin{cases} \left(\frac{2-\alpha}{4}\right) \beta_{q_t,h}^t + (1-\alpha)\beta_{q_t+1,h}^t \,, & q_t > 0 \\ \left(\frac{1-\alpha}{2}\right) \beta_{q_t,h}^t + (1-\alpha)\beta_{q_t+1,h}^t \,, & q_t = 0 \,, \end{cases}$$

and similarly for $\tilde{\beta}_{q_t+1,h}^{t+1}$ we have

$$\tilde{\beta}_{q_t+1,h}^{t+1} = \begin{cases} \left(\frac{2-\alpha}{4}\right) \tilde{\beta}_{q_t,h}^t + (1-\alpha)\tilde{\beta}_{q_t+1,h}^t \,, & q_t > 0 \\ \left(\frac{1-\alpha}{2}\right) \tilde{\beta}_{q_t,h}^t + (1-\alpha)\tilde{\beta}_{q_t+1,h}^t \,, & q_t = 0 \,. \end{cases}$$

For $q = q_t$ we have $\beta_{q_t,h}^{t+1} = 0$ and $\tilde{\beta}_{q_t,h}^{t+1} = 0$ if $q_t > 0$. All that remains is the update for $\beta_{q,h}^{t+1}$ and $\tilde{\beta}_{q,h}^{t+1}$ when $q = 0$, which can either correspond to the case that $q = q_t = 0$, or when $q \neq q_t > 0$. For these two cases, observe that $y_0 \in \mathcal{A}(u)$ only for $u = y_0$ and $u \in \{\tilde{y}_q \mid q \in \mathbb{N} \cup \{0\}\}$. From (5.36) we then

have

$$\begin{aligned}
\hat{\mu}_h^{t+1}(y_0) &= \hat{r}_h^t(y_0, y_0) + \frac{1}{\pi_h^{t+1}} \sum_{q \in \mathbb{N} \cup \{0\}} \hat{r}_h^t(\tilde{y}_q, y_0) \\
&= \hat{r}_h^t(y_0, y_0) + \frac{1}{\pi_h^{t+1}} \sum_{(u,u') \in \widetilde{\mathcal{W}} \times \mathcal{W}} \hat{r}_h^t(u, u') \\
&= \hat{r}_h^t(y_0, y_0) + \frac{\tilde{\xi}_h^t}{\pi_h^{t+1}} \\
&= r_t(y_0, y_0)\hat{\mu}_h^t(y_0) + \frac{\tilde{\xi}_h^t}{\pi_h^{t+1}} \\
&= r_t(y_0, y_0)\beta_{0,h}^t + \frac{\tilde{\xi}_h^t}{\pi_h^{t+1}} \\
&= [\![q_t > 0]\!](1-\alpha)\beta_{0,h}^t + \frac{\tilde{\xi}_h^t}{\pi_h^{t+1}} \,.
\end{aligned}$$

A similar argument can be used to show that

$$\hat{\mu}_h^{t+1}(\tilde{y}_0) = [\![q_t > 0]\!](1-\alpha)\tilde{\beta}_{0,h}^t + \xi_h^t \pi_h^{t+1} \,.$$

We therefore have for our update:

$$\beta_{0,h}^{t+1} = \begin{cases} (1-\alpha)\beta_{0,h}^t + \frac{\tilde{\xi}_h^t}{\pi_h^{t+1}} \,, & q_t > 0 \\ \frac{\tilde{\xi}_h^t}{\pi_h^{t+1}} \,, & q_t = 0 \,, \end{cases}$$

and similarly for $\tilde{\beta}_{0,h}^{t+1}$ we have

$$\tilde{\beta}_{0,h}^{t+1} = \begin{cases} (1-\alpha)\tilde{\beta}_{0,h}^t + \xi_h^t \pi_h^{t+1} \,, & q_t > 0 \\ \xi_h^t \pi_h^{t+1} \,, & q_t = 0 \,. \end{cases}$$

At last we can write the three routines, **initialize**, **get**, and **update** required by Algorithm 7 for our CGMC. These are given in routines 10, 11, and 12, respectively. Algorithm 7 with these three routines gives the AdaptLTM algorithm. Note that the first line of routine 12 is included in this routine for convenience, rather than re-writing Algorithm 7. We can, of course, compute

the tree height once per trial and cache this value. In Appendix D we give an algorithm to compute the tree-height function online in $\mathcal{O}(1)$ time per trial. With this algorithm for computing q_t, note that routines 10, 11, and 12 each require $\mathcal{O}(1)$ time per trial (per policy). The AdaptLTM algorithm therefore, somewhat remarkably, has a per-trial time complexity of $\mathcal{O}(N)$, which matches the time complexity of the algorithm of [2], and Algorithm 3 of Chapter 4, for the problem of non-adaptive LTM in the experts setting. Together with Algorithm 8 given in Appendix D to compute the tree height online, the space complexity of AdaptLTM is $\mathcal{O}(N \log t)$ on trial t.

In the next section, we present results from simulated experiments. When performing our experiments we found AdaptLTM to suffer from numerical instability, primarily from the quantity $\pi_h^t = \prod_{s=1}^{t-1} \psi_h^s$ quickly becoming very small for some policies. In Appendix C we give an alternative implementation that avoids having to compute this quantity, at the cost of a worse per-trial time complexity of $\mathcal{O}(N \log t)$ on trial t.

5.7 Experiments

In this section we present a comparison of AdaptLTM against several of the adaptive online algorithms considered in this thesis adapted to the setting of contextual bandits. These algorithms include the Fixed Share algorithm [30], the circadian specialist algorithm with simple Markov prior [2] (which we now refer to as "PBTS"), the projection analogue of Fixed Share, which we studied in Chapter 4 ("Proj FS") and the PoDS-θ algorithm of Chapter 4, which we showed is the projection analogue of PBTS. These algorithms are all adapted in the usual way to the contextual bandits setting. We also include the non-adaptive Exp4 algorithm.

We present the results of simulated experiments on synthetic data, where we simulate contextual bandits in the following manner. We set $A = 16$, and on each trial one action is *good* while the others are *bad*. The loss of the good action is drawn uniformly from $[0, 0.025]$, giving an expected loss of 0.0125

Routine 10 initialize(h)

$\pi_h^1 \leftarrow 1$
$\sigma_h^1 \leftarrow \alpha$
$\tilde{\sigma}_h^1 \leftarrow 1 - \alpha$
$\xi_h^1 \leftarrow \alpha^2$
$\tilde{\xi}_h^1 \leftarrow \frac{\alpha(1-\alpha)}{2}$
for $q \in \mathbb{N} \cup \{0\}$ **do**
    $\beta_{q,h}^1 := 0$
    $\tilde{\beta}_{q,h}^1 := 0$
end for
$\beta_{0,h}^1 \leftarrow \alpha$
$\tilde{\beta}_{1,h}^1 \leftarrow \frac{1-\alpha}{2}$

Routine 11 get(t, h)

return $(\sigma_h^t, \tilde{\sigma}_h^t)$

Routine 12 update(t, h)

$q_t \leftarrow \text{tree-height}(t)$
$\pi_h^{t+1} \leftarrow \psi_h^t \pi_h^t$
$\sigma_h^{t+1} \leftarrow \psi_h^t \sigma_h^t - \pi_h^{t+1} \xi_h^t + \tilde{\xi}_h^t$
$\tilde{\sigma}_h^{t+1} \leftarrow \tilde{\sigma}_h^t - \tilde{\xi}_h^t + \pi_h^{t+1} \xi_h^t$
if $q_t = 0$ **then**
    $\beta_{0,h}^{t+1} \leftarrow \tilde{\xi}_h^t / \pi_h^{t+1}$
    $\tilde{\beta}_{0,h}^{t+1} \leftarrow \xi_h^t \pi_h^{t+1}$
    $\beta_{1,h}^{t+1} \leftarrow (1-\alpha)\beta_{1,h}^t + \frac{(1-\alpha)}{2}\beta_{0,h}^t$
    $\tilde{\beta}_{1,h}^{t+1} \leftarrow (1-\alpha)\tilde{\beta}_{1,h}^t + \frac{(1-\alpha)}{2}\tilde{\beta}_{0,h}^t$
else
    $\beta_{0,h}^{t+1} \leftarrow (1-\alpha)\beta_{0,h}^t + \tilde{\xi}_h^t / \pi_h^{t+1}$
    $\tilde{\beta}_{0,h}^{t+1} \leftarrow (1-\alpha)\tilde{\beta}_{0,h}^t + \xi_h^t \pi_h^{t+1}$
    $\beta_{q_t,h}^{t+1} \leftarrow 0$
    $\tilde{\beta}_{q_t,h}^{t+1} \leftarrow 0$
    $\beta_{(q_t+1),h}^{t+1} \leftarrow (1-\alpha)\beta_{(q_t+1),h}^t + \frac{(2-\alpha)}{4}\beta_{q_t,h}^t$
    $\tilde{\beta}_{(q_t+1),h}^{t+1} \leftarrow (1-\alpha)\tilde{\beta}_{(q_t+1),h}^t + \frac{(2-\alpha)}{4}\tilde{\beta}_{q_t,h}^t$
end if
for $q \in \mathbb{N} \setminus \{q_t, q_t + 1\}$ **do**
    $\beta_{q,h}^{t+1} := \beta_{q,h}^t$
    $\tilde{\beta}_{q,h}^{t+1} := \tilde{\beta}_{q,h}^t$
end for
$\xi_h^{t+1} \leftarrow \xi_h^t + \alpha \sum_{q \in \{0, q_t, q_t+1\}} (\beta_{q,h}^{t+1} - \beta_{q,h}^t)$
$\tilde{\xi}_h^{t+1} \leftarrow \tilde{\xi}_h^t + \alpha \sum_{q \in \{0, q_t, q_t+1\}} (\tilde{\beta}_{q,h}^{t+1} - \tilde{\beta}_{q,h}^t)$

per trial, while the loss of any other action is drawn uniformly from $[0, 0.5]$, giving an expected loss of 0.25 per trial. On each trial, we generate $N = 1024$ policies, with one policy being *good*, and the remaining *bad*. The distribution of the good policy is parameterized by ϵ, and is concentrated with probability mass $1 - \epsilon$ on the current good action, and with mass $\frac{\epsilon}{A-1}$ uniformly on all other actions (we set $\epsilon = 0.1$). The distributions of all other policies are drawn uniformly from Δ_A on each trial.

We select a global pool of size $M = 24$ good policies uniformly at random and have $T = 21600$ trials. The sequence of trials is divided into $E = 18$ epochs, with each epoch being divided into 12 segments. For ease of visualization all epochs are of equal length, and all segments are of equal length. A good policy is then assigned to a given segment in the following manner. For each epoch i we uniformly sample $M^i = |\mathcal{P}^i| = 3$ policies from $\mathcal{P}$. Each segment of the epoch is then assigned one of these policies uniformly at random. Thus in epoch i we have $K^i \leq 12$ and $M^i \leq 3$. See Figure 5.10 for an example of such a trial sequence.

On each trial, the expert policies are revealed to the algorithms. After announcing its decision, each algorithm receives only the corresponding loss of the chosen action. For each algorithm, the learning rate η was tuned by selecting the optimal value from a line search over a range of values, as this gave universally better performance for all algorithms over using the theoretically optimal learning rates. The values of η for each algorithm are given in Appendix E. All other parameters were set to the optimal tuning suggested by the regret bounds of the algorithms with respect to the sequence designed in the experiments (see Table E.1 in Appendix E for these values, where it should be noted that for each algorithm we adopt the respective notation of the cited paper). Note that *exact* optimal values could not necessarily be given when these parameters depended specifically on characteristics of the sequence which were random variables due to the randomization of our experiments. In this case we used an upper bound on these characteristics. For

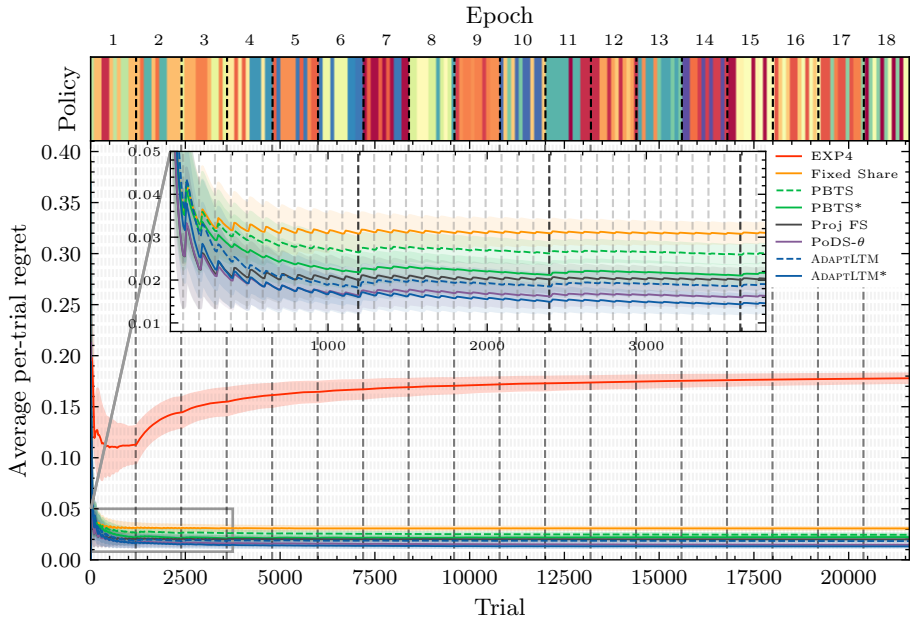


Figure 5.10: (Top) An example of an assignment of *good* policies (colors) across the trial sequence in our experiments. (Bottom) The per-trial (cumulative) regret of each algorithm with respect to the optimal sequence in hindsight over 21600 trials is shown. Error bands represent one standard deviation after 500 iterations. Light gray dashed lines represent segment borders. Dark gray dashed lines represent epoch borders. We observe adaptation on both the segment-level, as well as the epoch level.

example, if a parameter depended on the number of switches in the sequence, we used an upper bound on the maximum possible number of switches observed (in this case we have up to 12 segments per epoch, for 18 epochs, giving $K := (\sum_{i \in [E]} K^i) - 1 = 215$ switches). For AdaptLTM we set $\alpha = \frac{1}{4}$.

We also considered alternative tunings based on their empirical performance for the two circadian specialist algorithms (PBTS and AdaptLTM). For PBTS, we tried several tunings. We first tried both the original tuning suggested in [2], as well as the tuning suggested in our analysis in Chapter 4 (Theorem 15 and Proposition 24), keeping the tuning that performed better (in this case, the original tuning performed marginally better). We also considered setting "M" to 3, that is, the pool size in a single epoch, rather than $M = 24$, the global pool size. This tuning is denoted PBTS* in our experiments and is included as it showed significant improvement. For AdaptLTM, we also tried several

Table 5.1: Summary statistics of the total loss of each algorithm from our experiments, as well as the optimal sequence in hindsight. Mean total losses ± one standard deviation over 500 iterations are given.

Algorithm	Mean Total Loss ($\pm\sigma$)
EXP4	4624.64 ± 92.44
Fixed Share	1446.79 ± 16.58
PBTS	1311.4 ± 14.86
PBTS*	1263.56 ± 16.07
Proj FS	1215.72 ± 17.13
AdaptLTM	1179.12 ± 16.66
PoDS-θ	1100.47 ± 14.63
AdaptLTM*	1066.80 ± 15.73
Optimal Sequence	783.29 ± 12.18

tunings of α other than $1/4$, and include an example using $\alpha = 0.01$ as it also showed significant improvement. This tuning is denoted AdaptLTM*.

The average (cumulative) per-trial regret of the algorithms with respect to the optimal sequence in hindsight is shown in Figure 5.10. There are several interesting observations to be made. We first discuss the non-projection-based algorithms. Unsurprisingly, EXP4 performs the worst as it quickly overfits to the first well-performing policies in our sequence and cannot adapt to the changing sequence. Fixed Share is the worst-performing adaptive algorithm as it fails to exploit the reappearance of policies over time. The PBTS algorithm exploits reappearances over time, improving over Fixed Share, but does not adapt quickly across epochs. A significant improvement is observed with the alternate tuning (PBTS*). Conversely, AdaptLTM performs well across epochs compared to PBTS/PBTS*. On a local segment level we observe the adaptive behavior of the algorithms in the average per-trial regret as the algorithms learn the best policies. Similarly, on the global (epoch) level we see "shocks" to the average per-trial regret, exhibiting the adaptive behavior of the algorithms at the boundaries between epochs. This is most pronounced in PBTS, PoDS-θ, and AdaptLTM*, as they are forced to adapt to the change from pool $\mathcal{P}^i$ to $\mathcal{P}^{i+1}$. We see AdaptLTM* "recovering" quickly at these points, as these shocks are small.

Perhaps the most surprising observation is the strength of the performance of the projection-based algorithms, Proj FS and PoDS-θ. We observe that even PBTS* fails to improve over Proj FS, and AdaptLTM* only outperforms PoDS-θ with $\alpha = 0.01$, not the suggested $\alpha = 1/4$. We believe this is due to the nature of the projection-based updates (as discussed in Chapter 4) being much more *conservative* than the weight-sharing-styled updates of PBTS and AdaptLTM. Indeed, in Figure 5.10 we see that by the end of the first segment of the first epoch, both Proj FS and PoDS-θ are performing significantly better than the other algorithms. This can be explained by the algorithms' "adaptive updates" (projections) only ever changing weights when these weights are very small, unlike their weight-sharing counterparts (recall Theorem 20 from Chapter 4 where a guarantee is given due to such behavior). Thus in practice, if these updates change weights only rarely, then during the first segment the algorithms can learn the first policy much more quickly and "get ahead" of the other algorithms. During the first epoch, we see PoDS-θ and Proj FS "diverge" as PoDS-θ starts to outperform Proj FS by exploiting the reappearance of policies in the sequence. It is also during the first epoch that AdaptLTM*'s per-trial regret approaches that of PoDS-θ, and eventually we see AdaptLTM* outperforming PoDS-θ during the second and third epoch.

Overall this is very surprising to observe and raises the natural question: Is there a projection equivalent of AdaptLTM, just as PoDS-θ is the projection-equivalent of PBTS? As we have seen in Chapter 4, this question may be answered in the affirmative if there exists an MPP [1] mixing scheme which corresponds to the prior over circadian specialists introduced by our CGMC. We conjecture that such an MPP mixing scheme does exist, but deriving it seems particularly challenging, and we leave it as an interesting open problem.

Chapter 6

Conclusion

This thesis has studied the problem of adaptive online learning in several settings. In Chapter 3 we studied the problem of learning a sequence of binary graph labelings. Our primary result was an algorithm for predicting switching graph labelings with a per-trial prediction time of $O(\log n)$ and a mistake bound that *smoothly* tracks changes to the graph labeling over time. From a technical perspective, the most intriguing open problem is to eliminate the $\log \log T$ term from our bounds. The natural approach to this would be to replace the conservative *fixed-share* update with a *variable-share* update [30]; in our efforts, however, we found many technical problems with this approach. On both the more practical and speculative side, we observe that the specialists sets $\mathcal{B}_n$, and $\mathcal{F}_n$ were chosen to "prove bounds". In practice we can use any *hierarchical* graph clustering algorithm to produce a *complete* specialist set and furthermore, multiple such clusterings may be pooled. Such a pooled set of subgraph "motifs" could be then be used for example in a multi-task setting (see for example, [2]).

In Chapter 4 we considered the problem of switching within a small pool in the setting of prediction with expert advice. We gave an efficient projection-based algorithm for this problem, for which we proved the best known regret bound. We also gave an algorithm to compute relative entropy projection onto the simplex with non-uniform (lower) box constraints exactly in $\mathcal{O}(n)$ time, which may be of independent interest. We showed that the weight-sharing

equivalent of our projection-based algorithm is in fact a geometrically-decaying mixing scheme for *Mixing Past Posteriors* [1]. Furthermore, we showed that this mixing scheme corresponds exactly to the specialists algorithm with Markov prior [2] for this problem. We also proved a guarantee favoring projection updates over weight-sharing when updating weights may incur costs, such as in online portfolio selection with proportional transaction costs. Note that the work of [2] gave a Bayesian interpretation to MPP, however this is lost when one uses the projection update of PoDS. A natural question is whether there is also a Bayesian interpretation to these projection-based updates. We leave this as an interesting open problem.

Finally in Chapter 5 we introduced a refined model of switching with memory in which the trial sequence is divided into epochs, and the small pool changes between epochs. We presented results in both the full-information and partial-information settings, focusing on contextual bandits. We developed the algorithm AdaptLTM for this problem and proved a regret bound which essentially pays on a per-epoch basis rather than an average over all epochs. Our algorithm is computationally very efficient, having a time complexity of only $\mathcal{O}(N)$ per trial, which matches the speed of the existing algorithms for the much simpler problem of (non-adaptive) long-term memory. A current limitation of our approach in the contextual bandits setting is that the policies must be known in advance. An interesting future research direction would be to learn the policies online. Another future research direction is to extend the results in Chapter 4, in which we showed that the circadian specialists algorithm with simple Markov prior of [2] corresponded to MPP of [1] with a geometrically-decaying mixing scheme. Here we ask: what does the mixing scheme look like for AdaptLTM? This is a difficult question that we leave as an open problem. We note that solving this may also allow us to develop the projection-analogue of AdaptLTM, further extending the work of Chapter 4 and the PoDS methodology.

Appendix A

Proof of Theorem 14

We prove Theorem 14 by first proving a bound for the Kernel-Perceptron in an arbitrary Hilbert space before applying it to the Hilbert space described in the theorem. Given a semi-Hilbert space, $\mathcal{H}$, the kernel, $\boldsymbol{M}$, which describes this space induces the semi-norm $\|\boldsymbol{w}\|_{\boldsymbol{M}}^2 = \langle \boldsymbol{w}, \boldsymbol{w} \rangle_{\boldsymbol{M}} = \boldsymbol{w}^\top \boldsymbol{M}^+ \boldsymbol{w}$. We define the coordinate spanning set $\mathcal{V}_{\boldsymbol{M}} = \{\boldsymbol{v}_i := \boldsymbol{M}\boldsymbol{e}_i : i = 1, \ldots, n\}$, where $\boldsymbol{e}_i$ are the canonical basis vectors. For $\boldsymbol{w} \in \operatorname{span}(\mathcal{V}_{\boldsymbol{M}})$ we have

$$w_i = \boldsymbol{e}_i^\top \boldsymbol{M}\boldsymbol{M}^+ \boldsymbol{w} = \boldsymbol{v}_i^\top \boldsymbol{M}^+ \boldsymbol{w} = \langle \boldsymbol{v}_i, \boldsymbol{w} \rangle_{\boldsymbol{M}} \,, \tag{A.1}$$

which is simply the reproducing kernel property for kernel $\boldsymbol{M}$.

Recall that for the projection step of the algorithm we let the convex region Γ be the ball centered at the origin of radius γ as measured by the metric of the given Hilbert space.

We now present the following three lemmas, omitting the subscript $\|\cdot\|_{\boldsymbol{M}}$ when confusion does not arise. We assume there exists a sequence of target vectors $\boldsymbol{\mu}^1, \ldots \boldsymbol{\mu}^T$ such that $y^t \langle \boldsymbol{\mu}^t, \boldsymbol{x}^t \rangle_{\boldsymbol{M}} \geq 1$, $\forall t \in [T]$.

Lemma 29. *On all trials $t = 1, \ldots, T$,*

$$\|\boldsymbol{\mu}^t - \boldsymbol{w}^t\|^2 - \|\boldsymbol{\mu}^t - \dot{\boldsymbol{w}}^t\|^2 \geq \frac{[\![\hat{y}^t \neq y^t]\!]}{\max_t \|\boldsymbol{x}^t\|^2} \,. \tag{A.2}$$

Proof. The case where $\hat{y}^t = y^t$ is trivial, since no update is performed in this

case. For the case where $\hat{y}^t \neq y^t$, we include a *learning rate* parameter, η_t, to the Perceptron update rule such that $\dot{\boldsymbol{w}}^t = \boldsymbol{w}^t + \eta_t y^t \boldsymbol{x}^t$. We then have that

$$\begin{aligned}\|\boldsymbol{\mu}^t - \boldsymbol{w}^t\|^2 - \|\boldsymbol{\mu}^t - \dot{\boldsymbol{w}}^t\|^2 &= \|\boldsymbol{\mu}^t - \boldsymbol{w}^t\|^2 - \|\boldsymbol{\mu}^t - \boldsymbol{w}^t - \eta_t y^t \boldsymbol{x}^t\|^2 \\ &= 2\eta_t y^t \langle \boldsymbol{\mu}^t - \boldsymbol{w}^t, \boldsymbol{x}^t \rangle - \eta_t^2 (y^t)^2 \|\boldsymbol{x}^t\|^2 .\end{aligned}$$

Since we assume that $y^t \langle \boldsymbol{\mu}^t, \boldsymbol{x}^t \rangle \geq 1$ for all t, and it is clear that $y^t \langle \boldsymbol{w}^t, \boldsymbol{x}^t \rangle \leq 0$ when the algorithm makes a mistake, the first term can be lower bounded by $2\eta_t$, and since $(y^t)^2 = 1$ we have that

$$\|\boldsymbol{\mu}^t - \boldsymbol{w}^t\|^2 - \|\boldsymbol{\mu}^t - \dot{\boldsymbol{w}}^t\|^2 \geq 2\eta_t - \eta_t^2 \|\boldsymbol{x}^t\|^2 .$$

Setting $\eta_t = \frac{1}{\|\boldsymbol{x}^t\|^2}$ which maximizes the right-hand side completes the proof. □

Lemma 30. *On all trials $t = 1, \ldots, T-1$,*

$$\|\boldsymbol{\mu}^t - \dot{\boldsymbol{w}}^t\|^2 - \|\boldsymbol{\mu}^t - \boldsymbol{w}^{t+1}\|^2 \geq 0 . \tag{A.3}$$

Proof. This is an instance of the *generalized* Pythagorean Theorem for projection onto a convex set, since $\boldsymbol{w}^{t+1}$ is the projection of $\dot{\boldsymbol{w}}^t$ onto the set Γ. The proof is well known and we do not include it here (see e.g. [37, Theorem 2]). □

Lemma 31. *On all trials $t = 1, .., T-1$,*

$$\|\boldsymbol{\mu}^t - \boldsymbol{w}^{t+1}\|^2 - \|\boldsymbol{\mu}^{t+1} - \boldsymbol{w}^{t+1}\|^2 \geq \|\boldsymbol{\mu}^t\|^2 - \|\boldsymbol{\mu}^{t+1}\|^2 - 2\gamma \|\boldsymbol{\mu}^t - \boldsymbol{\mu}^{t+1}\| . \tag{A.4}$$

Proof. The case where $\boldsymbol{\mu}^t = \boldsymbol{\mu}^{t+1}$ is trivial. For the case where $\boldsymbol{\mu}^t \neq \boldsymbol{\mu}^{t+1}$, we have from the Cauchy-Schwarz inequality

$$\begin{aligned}\|\boldsymbol{\mu}^t - \boldsymbol{w}^{t+1}\|^2 - \|\boldsymbol{\mu}^{t+1} - \boldsymbol{w}^{t+1}\|^2 &= \|\boldsymbol{\mu}^t\|^2 - \|\boldsymbol{\mu}^{t+1}\|^2 - 2\langle \boldsymbol{\mu}^t - \boldsymbol{\mu}^{t+1}, \boldsymbol{w}^{t+1} \rangle \\ &\geq \|\boldsymbol{\mu}^t\|^2 - \|\boldsymbol{\mu}^{t+1}\|^2 - 2\|\boldsymbol{\mu}^t - \boldsymbol{\mu}^{t+1}\| \|\boldsymbol{w}^{t+1}\| .\end{aligned}$$

Finally the projection step of our algorithm necessarily upper-bounds $\|\boldsymbol{w}^{t+1}\|$ by γ, which completes the proof. □

Theorem 32. *For a Hilbert space $\mathcal{H}$ with norm induced by the kernel $\boldsymbol{M}$, let $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^T \in \mathcal{H}$ be a data sequence with $\max_{t\in[T]} \|\boldsymbol{x}^t\|_{\boldsymbol{M}}^2 =: R_{\boldsymbol{M}}$. Assume that there exists a sequence of vectors $\boldsymbol{u}^1, \ldots, \boldsymbol{u}^T \in \mathcal{H}$ such that $y^t \langle \boldsymbol{u}^t, \boldsymbol{x}^t \rangle_{\boldsymbol{M}} \geq 1$, for all $t \in [T]$, then the total number of mistakes for the switching Perceptron over T trials with $|\mathcal{K}| - 1$ switches is upper-bounded by*

$$M_P \leq R_{\boldsymbol{M}} \left(\|\boldsymbol{u}^{k_{|\mathcal{K}|}}\|_{\boldsymbol{M}}^2 + 2\gamma \sum_{i=1}^{|\mathcal{K}|-1} \|\boldsymbol{u}^{k_i} - \boldsymbol{u}^{k_{i+1}}\|_{\boldsymbol{M}} \right), \tag{A.5}$$

with $\mathcal{K}$ defined as in Theorem 4.

Proof. Summing equations (A.2), (A.3), and (A.4) for a given trial t leaves

$$\|\boldsymbol{\mu}^t - \boldsymbol{w}^t\|^2 - \|\boldsymbol{\mu}^{t+1} - \boldsymbol{w}^{t+1}\|^2 \geq \frac{[\![\hat{y}^t \neq y^t]\!]}{R_{\boldsymbol{M}}} + \|\boldsymbol{\mu}^t\|^2 - \|\boldsymbol{\mu}^{t+1}\|^2 - 2\gamma \|\boldsymbol{\mu}^t - \boldsymbol{\mu}^{t+1}\| . \tag{A.6}$$

By summing over trials $t = 1, \ldots, T$, the telescopic sums on both sides leave only

$$\|\boldsymbol{\mu}^1 - \boldsymbol{w}^1\|^2 - \|\boldsymbol{\mu}^{T+1} - \boldsymbol{w}^{T+1}\|^2 \geq \frac{M_P}{R_{\boldsymbol{M}}} + \|\boldsymbol{\mu}^1\|^2 - \|\boldsymbol{\mu}^{T+1}\|^2 - 2\gamma \sum_{i=1}^{|\mathcal{K}|-1} \|\boldsymbol{\mu}^{k_i} - \boldsymbol{\mu}^{k_{i+1}}\| , \tag{A.7}$$

where $M_P = \sum_{t=1}^{T} [\![\hat{y}^t \neq y^t]\!]$. Taking $\boldsymbol{w}^1 = \boldsymbol{0}$, and then re-arranging and dropping the negative squared terms then completes the proof. □

We now prove Theorem 14.

Proof. The proof follows straightforwardly from Theorem 32 with kernel $\boldsymbol{K} = \boldsymbol{L}_{\mathcal{G}}^{+} + R_L \boldsymbol{1}\boldsymbol{1}^\top$, using the fact that $R_{\boldsymbol{K}} = 2R_{\boldsymbol{L}} \leq 2R_{\mathcal{G}}$, and for $\boldsymbol{\mu} \in \{-1, 1\}^n$, $\|\boldsymbol{\mu}\|_{\boldsymbol{K}}^2 \leq 4\Phi_{\mathcal{G}}(\boldsymbol{\mu}) + \frac{1}{R_L}$. This gives

$$\gamma = \max_{k\in K} \|\boldsymbol{\mu}^k\|_{\boldsymbol{K}} \leq \sqrt{4\hat{\Phi} + \frac{1}{R_L}} , \tag{A.8}$$

and

$$\begin{aligned}
\|\boldsymbol{\mu}^{k_i} - \boldsymbol{\mu}^{k_{i+1}}\|_{\boldsymbol{K}}^2 &= \|\boldsymbol{\mu}^{k_i}\|_{\boldsymbol{K}}^2 + \|\boldsymbol{\mu}^{k_{i+1}}\|_{\boldsymbol{K}}^2 - 2\langle \boldsymbol{\mu}^{k_i}, \boldsymbol{\mu}^{k_{i+1}} \rangle_{\boldsymbol{K}} \\
&\leq \|\boldsymbol{\mu}^{k_i}\|_{\boldsymbol{K}}^2 + \|\boldsymbol{\mu}^{k_{i+1}}\|_{\boldsymbol{K}}^2 + 2|\langle \boldsymbol{\mu}^{k_i}, \boldsymbol{\mu}^{k_{i+1}} \rangle_{\boldsymbol{K}}| \\
&\leq \|\boldsymbol{\mu}^{k_i}\|_{\boldsymbol{K}}^2 + \|\boldsymbol{\mu}^{k_{i+1}}\|_{\boldsymbol{K}}^2 + 2\|\boldsymbol{\mu}^{k_i}\|_{\boldsymbol{K}} \|\boldsymbol{\mu}^{k_{i+1}}\|_{\boldsymbol{K}} \\
&= 4\Phi^{k_i} + 4\Phi^{k_{i+1}} + \frac{2}{R_L} + 2\sqrt{4\Phi^{k_i} + \frac{1}{R_L}}\sqrt{4\Phi^{k_{i+1}} + \frac{1}{R_L}} \\
&\leq 4\Phi^{k_i} + 4\Phi^{k_{i+1}} + \frac{2}{R_L} + 2\left(4\max\left\{\Phi^{k_i}, \Phi^{k_{i+1}}\right\} + \frac{1}{R_L}\right) \\
&\leq 16\max\left\{\Phi^{k_i}, \Phi^{k_{i+1}}\right\} + \frac{4}{R_L},
\end{aligned}$$

where the second inequality uses the Cauchy-Schwarz Inequality. Thus

$$\|\boldsymbol{\mu}^{k_i} - \boldsymbol{\mu}^{k_{i+1}}\|_{\boldsymbol{K}} \leq 2\sqrt{4\max\left\{\Phi^{k_i}, \Phi^{k_{i+1}}\right\} + \frac{1}{R_L}}. \tag{A.9}$$

Substituting these terms into (A.5) gives the following bound

$$M_P \leq 8R_L\Phi^{k_{|K|}} + 8\sqrt{4R_L\hat{\Phi} + 1}\sum_{i=1}^{|\mathcal{K}|-1}\sqrt{4R_L\max\left\{\Phi^{k_i}, \Phi^{k_{i+1}}\right\} + 1}.$$

Using the fact that $R_L \leq R_{\mathcal{G}}$, as well as $\Phi^{k_{|K|}} \leq \sqrt{\Phi^{k_{|K|}}\hat{\Phi}}$, and $\sum_{i=1}^{|\mathcal{K}|-1}\sqrt{\max\left\{\Phi^{k_i}, \Phi^{k_{i+1}}\right\}} \leq \sum_{i=1}^{|\mathcal{K}|}\sqrt{2\Phi^{k_i}}$ then gives

$$M_P \leq \mathcal{O}\left(R_{\mathcal{G}}\sum_{i=1}^{|\mathcal{K}|}\sqrt{\hat{\Phi}\Phi^{k_i}}\right),$$

as required. □

Appendix B

Proof of Theorem 26

Proof. We introduce several quantities which allow us to describe the following characterisation of a circadian pattern $\boldsymbol{\gamma} \in \{0,1\}^T$. For convenience, given $\boldsymbol{\gamma}$ we extend this vector by defining $\gamma_0 = \gamma_{T+1} = 0$. We then define

$$\kappa(\boldsymbol{\gamma}) := \sum_{t=1}^{T} [\![\gamma_t = 1 \wedge \gamma_{t-1} = 0]\!] , \tag{B.1}$$

which is the number of segments of "all ones" in $\boldsymbol{\gamma}$. We define $\tilde{\nu}_0(\boldsymbol{\gamma}) := 0$, and for all $j \in [\kappa(\boldsymbol{\gamma})]$ we recursively define:

$$\nu_j(\boldsymbol{\gamma}) := \min \{t \in [T] \mid t > \tilde{\nu}_{j-1}(\boldsymbol{\gamma}) \wedge \gamma_t = 1\} , \tag{B.2}$$

which is simply the trial on which the j^{th} "all ones" segment starts, and similarly

$$\tilde{\nu}_j(\boldsymbol{\gamma}) := \min \{t \in [T+1] \mid t > \nu_j(\boldsymbol{\gamma}) \wedge \gamma_t = 0\} ,$$

which is the trial on which the "all zeros" segment following the j^{th} all-ones segment starts. We will also require the length of each segment. For $j \in [\kappa(\boldsymbol{\gamma})]$, let

$$\varphi_j(\boldsymbol{\gamma}) := \tilde{\nu}_j(\boldsymbol{\gamma}) - \nu_j(\boldsymbol{\gamma})$$

denote the lengths of the "all-ones" segments, and let

$$\tilde{\varphi}_j(\boldsymbol{\gamma}) := \nu_j(\boldsymbol{\gamma}) - \tilde{\nu}_{j-1}(\boldsymbol{\gamma})$$

denote the lengths of the "all-zeros" segments. We finally define $\tilde{\varphi}_{\kappa(\boldsymbol{\gamma})+1}(\boldsymbol{\gamma}) := T + 2 - \tilde{\nu}_{\kappa(\boldsymbol{\gamma})}(\boldsymbol{\gamma})$ which is the length of the last segment of "all zeros".

The proof requires us to bound $-\ln(f(\boldsymbol{c}^h))$ for $h \in \mathcal{P}$. We thus bound the value of $-\ln f(\boldsymbol{\gamma})$ for any arbitrary $\boldsymbol{\gamma} \in \{0,1\}^T$, where $f(\boldsymbol{\gamma})$ is defined in (5.25).

Given some $\boldsymbol{\gamma} \in \{0,1\}^T$, we define the sequence $\hat{\boldsymbol{q}} := (\hat{q}_0, \hat{q}_1, \ldots, \hat{q}_{T+1})$ recursively as follows. Let $\hat{q}_0 := 0$, then for all $t \in [T] \cup \{0\}$ we define $\hat{q}_{t+1}$ as follows:

$$\hat{q}_{t+1} := \begin{cases} 0 & \gamma_{t+1} \neq \gamma_t \\ \hat{q}_t & \gamma_{t+1} = \gamma_t \text{ and } q_t \neq \hat{q}_t \\ \hat{q}_t + 1 & \gamma_{t+1} = \gamma_t \text{ and } q_t = \hat{q}_t \,. \end{cases}$$

Observe that since a change ($\gamma_{t+1} \neq \gamma_t$) corresponds to a particle in our CGMC transitioning to y_0 or $\tilde{y}_0$ (i.e., waking up or falling asleep), and when $q_t = q$ we have $r_t(y_q, y_q) = 0$ and $r_t(y_q, y_{q+1}) \neq 0$, then the sequence $\hat{\boldsymbol{q}}$ simply keeps track of the "depth" of a particle as it traverses our CGMC as defined in Section 5.6.1.

Given $\boldsymbol{\gamma}$ and its corresponding sequence $\hat{\boldsymbol{q}}$, we now define the sequence $\boldsymbol{u} = (u_0, u_1, \ldots, u_T) \in \mathcal{Y}^{T+1}$, where

$$u_t := \begin{cases} \tilde{y}_{\hat{q}_t} & \gamma_t = 0\,, \\ y_{\hat{q}_t} & \gamma_t = 1\,. \end{cases}$$

That is, $\boldsymbol{u}$ is the sequence of states the particle is in during its trajectory. Clearly $u_t \in \mathcal{W}$ if and only if $\gamma_t = 1$ and therefore (ignoring the components u_0 and u_{T+1}) we have $\boldsymbol{u} \in \Gamma(\boldsymbol{\gamma})$. By (5.23) and (5.25) we then have

$$f(\boldsymbol{\gamma}) = \sum_{\boldsymbol{z} \in \Gamma(\boldsymbol{\gamma})} \hat{w}(\boldsymbol{z}) \geq \hat{w}(\boldsymbol{u}) = \iota(u_1) \prod_{t=1}^{T-1} r_t(u_t, u_{t+1})\,. \tag{B.3}$$

Now for simplicity we extend our transition function by defining

$$r_0(\tilde{y}_0, y_0) := \alpha$$

and

$$r_0(\tilde{y}_0, \tilde{y}_1) := [\![q_0 = 0]\!](1-\alpha)\,.$$

Now since $q_0 = 0$ by definition then we have $r_0(\tilde{y}_0, \tilde{y}_1) = 1-\alpha$, and therefore $\iota(u_1) = r_0(u_0, u_1))$ which when substituted into (B.3) gives

$$f(\boldsymbol{\gamma}) \geq \prod_{t=0}^{T-1} r_t(u_t, u_{t+1})\,.$$

We will now split this product up into several terms, firstly for $j \in [\kappa(\boldsymbol{\gamma})]$ we define

$$f_j := \prod_{t=\nu_j(\boldsymbol{\gamma})}^{\tilde{\nu}_j(\boldsymbol{\gamma})-1} r_t(u_t, u_{t+1})$$

for the "all-ones" segments. Similarly we define

$$\tilde{f}_j := \prod_{t=\tilde{\nu}_{j-1}(\boldsymbol{\gamma})}^{\nu_j(\boldsymbol{\gamma})-1} r_t(u_t, u_{t+1})$$

for the "all-zeros" segments. We finally define

$$\tilde{f}_{\kappa(\boldsymbol{\gamma})+1} := \prod_{t=\tilde{\nu}_{\kappa(\boldsymbol{\gamma})}(\boldsymbol{\gamma})}^{T-1} r_t(u_t, u_{t+1})\,.$$

Substituting these terms into (B.3) and taking logarithms gives

$$-\ln f(\boldsymbol{\gamma}) \leq -\sum_{j\in[\kappa(\boldsymbol{\gamma})]} \ln(f_j) - \sum_{j\in[\kappa(\boldsymbol{\gamma})+1]} \ln(\tilde{f}_j)\,. \tag{B.4}$$

We now bound $-\ln(f_j)$ for some fixed $j \in [\kappa(\boldsymbol{\gamma})]$. First, let $q^* := \hat{q}_{\tilde{\nu}_j(\boldsymbol{\gamma})-1}$ be the maximum depth reached in the CGMC during that segment. Then for all $q \in [q^*] \cup \{0\}$ define

$$m_q := \min\{s \in [\nu_j(\boldsymbol{\gamma}), \tilde{\nu}_j(\boldsymbol{\gamma})-1] \mid \hat{q}_s = q\}\,,$$

be the first trial of the segment that our particle reached state y_q. Since for all

$s \in [\nu_j(\boldsymbol{\gamma}), \tilde{\nu}_j(\boldsymbol{\gamma}) - 1]$ we have $\gamma_s = 1$, then $\hat{q}_{\nu_j(\boldsymbol{\gamma})+s-1}$ is monotonic increasing over $s \in [\varphi_j(\boldsymbol{\gamma})]$. This implies that the sequence $(m_0, m_1, \ldots, m_{q^*})$ is monotonic increasing.

Note that we have

$$f_j = \prod_{t=\nu_j(\boldsymbol{\gamma})}^{\tilde{\nu}_j(\boldsymbol{\gamma})-1} r_t(u_t, u_{t+1}) = \left(\prod_{q=0}^{q^*-1} \prod_{s=m_q}^{m_{q+1}-1} r_s(u_s, u_{s+1}) \right) \prod_{t=m_{q^*}}^{\tilde{\nu}_j(\boldsymbol{\gamma})-1} r_t(u_t, u_{t+1}) \,,$$

and thus we now bound each of these products individually. We start with $\prod_{s=m_0}^{m_1-1} r_s(u_s, u_{s+1})$.

Note that since $q_t = 0$ whenever t is even, we have $m_1 - m_0 \leq 2$. We then have two cases.

Case 1: $(m_1 = m_0 + 1)$ This case implies that $q_{m_0} = 0$ and so $r_{m_0}(u_{m_0}, u_{m_1}) = r_{m_0}(y_0, y_1) = 1 - \alpha$.

Case 2: $(m_1 = m_0 + 2)$ If $m_1 = m_0 + 2$, then $q_{m_0} \neq 0$, so $r_{m_0}(y_0, y_0) = 1 - \alpha$ and $q_{m_0+1} = 0$ so $r_{m_0+1}(y_0, y_1) = 1 - \alpha$.

In either case we have

$$\prod_{s=m_0}^{m_1-1} r_s(u_s, u_{s+1}) \geq (1 - \alpha)^2 \,. \tag{B.5}$$

We now bound the term $\prod_{s=m_q}^{m_{q+1}-1} r_s(u_s, u_{s+1})$ for $q \in [q^* - 1]$. Observe that we have two cases, either $m_{q+1} = m_q + 2^{q-1}$ or $m_{q+1} = m_q + 2^q + 2^{q-1}$.

Case 1: $(m_{q+1} = m_q + 2^{q-1})$ For this case we will take the inductive hypothesis that for all $s \in [2^{q-1}] \cup \{0\}$ we have

$$\prod_{t=m_q}^{m_q+s-1} r_t(u_t, u_{t+1}) = 1 - \frac{\alpha s}{2^q} \,.$$

First, note that this holds for $s = 0$, since the product over all elements of the empty set is equal to unity. Now suppose it holds for some $s \in [2^{q-1} - 1] \cup \{0\}$.

Note that we have

$$\begin{aligned} k_{(m_q+s),q} &:= m_q + s - \max\{t \in [m_q + s] \mid q_{t-1} = q - 1\} \\ &= m_q + s - m_q \\ &= s\,, \end{aligned} \tag{B.6}$$

and by the inductive hypothesis we have

$$\prod_{t=m_q}^{m_q+s} r_t(u_t, u_{t+1}) = r_{m_q+s}(u_{m_q+s}, u_{m_q+s+1}) \left(1 - \frac{\alpha s}{2^q}\right). \tag{B.7}$$

For all such s we have $\hat{q}_{m_q+s} = q$ (and thus $u_{m_q+s} = y_q$). Indeed, observe that when $s < 2^{q-1} - 1$ we have $\hat{q}_{m_q+s} = q$ and $\hat{q}_{m_q+s+1} = q$, and when $s = 2^{q-1} - 1$ we have $q_{m_q+s} = q$ and $\hat{q}_{m_q+s+1} = q + 1$ (and thus $u_{m_q+s+1} = y_{q+1}$). In either case we have, by (B.6) that

$$\begin{aligned} r_{m_q+s}(u_{m_q+s}, u_{m_q+s+1}) &= 1 - \omega_{(m_q+s),q} \\ &= 1 - \frac{\alpha}{2^q - \alpha k_{(m_q+s),q}} \\ &= 1 - \frac{\alpha}{2^q - \alpha s}\,. \end{aligned}$$

Substituting this into (B.7) gives

$$\begin{aligned} \prod_{t=m_q}^{m_q+(s+1)-1} r_t(u_t, u_{t+1}) &= \left(1 - \frac{\alpha}{2^q - \alpha s}\right)\left(1 - \frac{\alpha s}{2^q}\right) \\ &= \left(\frac{2^q - \alpha(s+1)}{2^q - \alpha s}\right)\left(\frac{2^q - \alpha s}{2^q}\right) \\ &= 1 - \frac{\alpha(s+1)}{2^q}\,, \end{aligned}$$

so the inductive hypothesis holds for $s+1$ and hence holds for all $s \in [2^{q-1}] \cup \{0\}$. Specifically it holds for $s = 2^{q-1}$ which gives

$$\begin{aligned}\prod_{t=m_q}^{m_{q+1}-1} r_t(u_t, u_{t+1}) &= \prod_{t=m_q}^{m_q+2^{q-1}-1} r_t(u_t, u_{t+1}) \\ &= 1 - \frac{\alpha 2^{q-1}}{2^q} \\ &= 1 - \frac{\alpha}{2}\,.\end{aligned}$$

Case 2: $(m_{q+1} = m_q + 2^q + 2^{q-1})$ Using the same argument as in Case 1, it can be shown that for this case we have

$$\prod_{t=m_q}^{m_q+2^q-1} r_t(u_t, u_{t+1}) = 1 - \frac{\alpha 2^q}{2^q} = 1 - \alpha\,,$$

and

$$\prod_{t=m_q+2^q}^{m_q+2^q+2^{q-1}-1} r_t(u_t, u_{t+1}) = 1 - \frac{\alpha 2^{q-1}}{2^q} = 1 - \frac{\alpha}{2}\,,$$

so

$$\begin{aligned}\prod_{t=m_q}^{m_{q+1}-1} r_t(u_t, u_{t+1}) &= \left(\prod_{t=m_q}^{m_q+2^q-1} r_t(u_t, u_{t+1})\right)\left(\prod_{t=m_q+2^q}^{m_q+2^q+2^{q-1}-1} r_t(u_t, u_{t+1})\right) \\ &= (1-\alpha)\left(1 - \frac{\alpha}{2}\right)\,.\end{aligned}$$

In either case we have

$$\prod_{t=m_q}^{m_{q+1}-1} r_t(u_t, u_{t+1}) \geq (1-\alpha)\left(1 - \frac{\alpha}{2}\right)\,. \tag{B.8}$$

The same argument can be used to show that

$$\prod_{t=m_{q^*}}^{\tilde{\nu}_j(\boldsymbol{\gamma})-2} r_t(u_t, u_{t+1}) \geq (1-\alpha)\left(1 - \frac{\alpha}{2}\right)\,. \tag{B.9}$$

Now, since $u_{(\tilde{\nu}_j(\boldsymbol{\gamma})-1)} = y_{q^*}$ and $u_{\tilde{\nu}_j(\boldsymbol{\gamma})} = \tilde{y}_0$ we have

$$\begin{aligned} r_{(\tilde{\nu}_j(\boldsymbol{\gamma})-1)}(u_{(\tilde{\nu}_j(\boldsymbol{\gamma})-1)}, u_{\tilde{\nu}_j(\boldsymbol{\gamma})}) &= r_{(\tilde{\nu}_j(\boldsymbol{\gamma})-1)}(y_{q^*}, \tilde{y}_0) \\ &= \omega_{(\tilde{\nu}_j(\boldsymbol{\gamma})-1),q^*} \\ &= \frac{\alpha}{2^{q^*} - \alpha k_{(\tilde{\nu}_j(\boldsymbol{\gamma})-1),q^*}} \\ &\geq \frac{\alpha}{2^{q^*}} \,. \end{aligned} \tag{B.10}$$

Combining (B.5), (B.8), and (B.9), and (B.10) then gives

$$\begin{aligned} f_j &= \prod_{t=\nu_j(\boldsymbol{\gamma})}^{\tilde{\nu}_j(\boldsymbol{\gamma})-1} r_t(u_t, u_{t+1}) \\ &= r_{(\tilde{\nu}_j(\boldsymbol{\gamma})-1)}(u_{(\tilde{\nu}_j(\boldsymbol{\gamma})-1)}, u_{\tilde{\nu}_j(\boldsymbol{\gamma})}) \prod_{t=\nu_j(\boldsymbol{\gamma})}^{\tilde{\nu}_j(\boldsymbol{\gamma})-2} r_t(u_t, u_{t+1}) \\ &= r_{(\tilde{\nu}_j(\boldsymbol{\gamma})-1)}(u_{(\tilde{\nu}_j(\boldsymbol{\gamma})-1)}, u_{\tilde{\nu}_j(\boldsymbol{\gamma})}) \left(\prod_{q=0}^{q^*-1} \prod_{s=m_q}^{m_{q+1}-1} r_s(u_s, u_{s+1}) \right) \prod_{t=m_{q^*}}^{\tilde{\nu}_j(\boldsymbol{\gamma})-2} r_t(u_t, u_{t+1}) \\ &\geq \frac{\alpha}{2^{q^*}} (1-\alpha)^2 \left((1-\alpha) \left(1 - \frac{\alpha}{2} \right) \right)^{q^*} , \end{aligned}$$

and therefore

$$\ln(f_j) \geq \ln(\alpha) + 2\ln(1-\alpha) + q^* \left(\ln(1-\alpha) + \ln\left(1 - \frac{\alpha}{2}\right) - \ln(2) \right) . \tag{B.11}$$

Since $(\ln(1-\alpha) + \ln(1-\alpha/2) - \ln(2))$ is negative for $\alpha \in [0,1]$, in order to lower bound $\ln(f_j)$ we now upper bound q^* in terms of $\varphi_j(\boldsymbol{\gamma})$, the length of the segment. Assume without loss of generality that $q^* > 0$. Recall that for all $q \in [q^* - 1]$ we have $q_{m_q - 1} = q - 1$ and $q_{m_{q+1}-1} = q$, and hence

$$m_{q+1} - m_q \in \{2^{q-1}, 2^q + 2^{q-1}\} .$$

Similarly we have

$$m_1 - \nu_j(\boldsymbol{\gamma}) = m_1 - m_0 \in \{1, 2\} ,$$

and

$$\tilde{\nu}_j(\boldsymbol{\gamma}) - m_{q^*} \in [1, 2^{q^*} + 2^{q^*-1}] .$$

We therefore have

$$\varphi_j(\boldsymbol{\gamma}) = \tilde{\nu}_j(\boldsymbol{\gamma}) - \nu_j(\boldsymbol{\gamma}) \geq 1 + \sum_{q \in [q^*-1]} 2^{q-1} + 1 = 2 + (2^{q^*} - 1) > 2^{q^*} ,$$

so

$$q^* \leq \log_2 (\varphi_j(\boldsymbol{\gamma})) = \frac{\ln (\varphi_j(\boldsymbol{\gamma}))}{\ln (2)} ,$$

which, when substituted into (B.11) gives

$$\ln (f_j) \geq \ln (\alpha) + 2 \ln (1 - \alpha) + \frac{\ln (1 - \alpha) + \ln \left(1 - \frac{\alpha}{2}\right) - \ln (2)}{\ln (2)} \ln (\varphi_j(\boldsymbol{\gamma})) .$$

When $\alpha = 1/4$ we have

$$- \ln (\alpha) - 2 \ln (1 - \alpha) < 2 ,$$

and

$$- \frac{\ln (1 - \alpha) + \ln \left(1 - \frac{\alpha}{2}\right) - \ln (2)}{\ln (2)} < 2 ,$$

so

$$- \ln (f_j) \leq 2 \ln (e \varphi_j(\boldsymbol{\gamma})) . \tag{B.12}$$

The same argument can be used to show that similarly

$$- \ln (\tilde{f}_j) \leq 2 \ln (e \tilde{\varphi}_j(\boldsymbol{\gamma})) . \tag{B.13}$$

and

$$- \ln (\tilde{f}_{\kappa(\boldsymbol{\gamma})+1}) \leq 2 \ln (e \tilde{\varphi}_{\kappa(\boldsymbol{\gamma})+1}(\boldsymbol{\gamma})) . \tag{B.14}$$

We now substitute (B.12), (B.13), and (B.14) into (B.4) gives

$$- \ln (f(\boldsymbol{\gamma})) \leq \sum_{j \in [\kappa(\boldsymbol{\gamma})]} 2 \ln (e \varphi_j(\boldsymbol{\gamma})) + \sum_{j \in [\kappa(\boldsymbol{\gamma})+1]} 2 \ln (e \tilde{\varphi}_j(\boldsymbol{\gamma})) . \tag{B.15}$$

We now turn our attention to bounding $\bar{C}$ from (5.19) using (B.15). We first introduce the following definitions. Let the set of trials on which a switch occurs be denoted as

$$\mathcal{Z} := \{t \in [2, T] \mid h_{t-1} \neq h_t\} \cup \{1\} .$$

For each switching point $t \in \mathcal{Z}$ we define

$$g^t := \min \{s \in [T] \mid s > t \wedge h_s \neq h_t\} - t ,$$

which is the length of that segment. Here we define the minimiser of the empty set to be equal to $T + 1$. We similarly define

$$\tilde{g}^t := t - \max \{s \in [T] \mid s < t \wedge h_{s-1} = h_t\} ,$$

that is, the number of trials since the last time h_t was the relevant policy. Here we define the maximiser of the empty set to be equal to zero. Finally for all $h \in \mathcal{P}$ we define

$$\hat{g}(h) := T + 1 - \max \{s \in [T] \mid h_s = h\} ,$$

that is, the number of trials which occur after the last time policy h is observed in the sequence $\boldsymbol{h}$.

Recall the definition of $\boldsymbol{c}^h$ given in (5.8): that for given policy sequence $\boldsymbol{h} \in \mathcal{H}^T$, for all $t \in [T]$ we have $c_t^h := [\![h_t = h]\!]$. For convenience we define $c_0^h := 0$ such that for all $t \in [T]$ we have $[\![c_t^h = 1 \wedge c_{t-1}^h = 0]\!] = [\![t \in \mathcal{Z} \wedge h_t = h]\!]$. Note that from (B.1) this means that

$$\kappa(\boldsymbol{c}^h) = \sum_{t=1}^{T} [\![c_t^h = 1 \wedge c_{t-1}^h = 0]\!] = \sum_{t \in \mathcal{Z}} [\![h_t = h]\!] ,$$

and from (B.2) that

$$\{t \in \mathcal{Z} \mid h_t = h\} = \{\nu_j(\boldsymbol{c}^h) \mid j \in [\kappa(\boldsymbol{c}^h)]\}\,. \tag{B.16}$$

Note that for all $j \in [\kappa(\boldsymbol{c}^h)]$ we have, by definition of $\tilde{\nu}_{j-1}(\boldsymbol{c}^h)$ and (5.8) that

$$\begin{aligned}\tilde{\nu}_{j-1}(\boldsymbol{c}^h) &= \max\{t \in [T] \mid t < \nu_j(\boldsymbol{c}^h) \wedge c^h_{t-1} = 1\} \\ &= \max\{t \in [T] \mid t < \nu_j(\boldsymbol{c}^h) \wedge h_{t-1} = h\}\,,\end{aligned}$$

and that, by definition of $\tilde{\nu}_j(\boldsymbol{c}^h)$ and (5.8) that

$$\begin{aligned}\tilde{\nu}_j(\boldsymbol{c}^h) &= \min\{t \in [T+1] \mid t > \nu_j(\boldsymbol{c}^h) \wedge c^h_t = 0\} \\ &= \min\{t \in [T+1] \mid t > \nu_j(\boldsymbol{c}^h) \wedge h_t \neq h\}\,.\end{aligned}$$

These two equations give for all $j \in [\kappa(\boldsymbol{c}^h)]$,

$$\tilde{\varphi}_j(\boldsymbol{c}^h) = \nu_j(\boldsymbol{c}^h) - \tilde{\nu}_{j-1}(\boldsymbol{c}^h) = \tilde{g}^{\nu_j(\boldsymbol{c}^h)}\,, \tag{B.17}$$

and

$$\varphi_j(\boldsymbol{c}^h) = \tilde{\nu}_j(\boldsymbol{c}^h) - \nu_j(\boldsymbol{c}^h) = g^{\nu_j(\boldsymbol{c}^h)}\,, \tag{B.18}$$

and finally

$$\begin{aligned}\tilde{\varphi}_{\kappa(\boldsymbol{c}^h)+1}(\boldsymbol{c}^h) &= T + 2 - \tilde{\nu}_{\kappa(\boldsymbol{c}^h)}(\boldsymbol{c}^h) \\ &= T + 1 - \max\{s \in [T] \mid h_s = h\} \\ &= \hat{g}(h)\,.\end{aligned} \tag{B.19}$$

Substituting (B.17), (B.18), and (B.19) into (B.15) gives

$$-\ln(f(\boldsymbol{c}^h)) \leq \sum_{j \in [\kappa(\boldsymbol{c}^h)]} 2\ln(e g^{\nu_j(\boldsymbol{c}^h)}) + \sum_{j \in [\kappa(\boldsymbol{c}^h)]} 2\ln(e \tilde{g}^{\nu_j(\boldsymbol{c}^h)}) + 2\ln(e\hat{g}(h))\,,$$

and it then follows from (B.16) that

$$\sum_{h\in\mathcal{P}} -\ln\left(f(\boldsymbol{c}^h)\right) \leq 2\sum_{t\in\mathcal{Z}}\left(\ln\left(eg^t\right)+\ln\left(e\tilde{g}^t\right)\right)+2\sum_{h\in\mathcal{P}}\ln\left(e\hat{g}(h)\right). \tag{B.20}$$

We now bound $\bar{C}$ using (B.20) in terms of our adaptive LTM model. Recall that the model is defined by a partitioning of the trial sequence into E epochs $\mathcal{E} := (e^1, \ldots, e^E) \subseteq [T]$ where $1 = e^1 < e^2 < \ldots < e^E \leq T$ and conventionally we have $e^{E+1} := T + 1$.

For all $i \in [E]$ the i^{th} epoch corresponds to the segment of trials $\mathcal{Q}^i := [e^i, e^{i+1} - 1]$, noting that the set $\{\mathcal{Q}^i \mid i \in [E]\}$ partitions $[T]$. For all $i \in [E]$ recall that $T^i := e^{i+1} - e^i$, $\mathcal{P}^i := \{h \in \mathcal{P} \mid \exists t \in \mathcal{Q}^i : h_t = h\}$, and $M^i := |\mathcal{P}^i|$. We also define the set of switching points for epoch i as

$$\mathcal{Z}^i := \{t \in \mathcal{Q}^i \mid h_{t-1} \neq h_t\} \cup \{e^i\},$$

such that $K^i = |\mathcal{Z}^i|$.

For convenience we also define $\mathcal{P}^0 := \mathcal{P}$ and $\mathcal{P}^{E+1} := \mathcal{P}$, and for all $i \in [E+1]$ we define $\Psi^i := \mathcal{P}^i \setminus \mathcal{P}^{i-1}$, and

$$\Phi := \sum_{i\in[E-1]} |\mathcal{P}^i \vartriangle \mathcal{P}^{i+1}| = \sum_{i\in[E-1]} \left(|\mathcal{P}^{i+1} \setminus \mathcal{P}^i| + |\mathcal{P}^i \setminus \mathcal{P}^{i+1}|\right).$$

For $i \in [E+1]$ we now define three quantities, d_i^t, $\tilde{d}_i^t$, and $\hat{d}_i(h)$. These quantities are analogous to the quantities g^t, $\tilde{g}^t$, and $\hat{g}(h)$, but are defined with respect to a given epoch, rather than the global trial sequence. Specifically, given $i \in [E]$ and $t \in \mathcal{Z}^i$ we define

$$d_i^t := \min\{s \in \mathcal{Q}^i \mid s > t \wedge h_s \neq h_t\} - t$$

to be the length of the policy's segment in the epoch. Note that here we define

the minimiser of the empty set is defined to be equal to e^{i+1}. We define

$$\tilde{d}_i^t := t - \max\{s \in \mathcal{Q}^i \mid s < t \wedge h_{s-1} = h_t\},$$

to be the number of trials since the last time h_t was the relevant policy in that epoch, where the maximiser of the empty set is defined to be equal to $e^i - 1$. Finally, for all $h \in \mathcal{P}^i$ we define

$$\hat{d}_i(h) := e^{i+1} - \max\{s \in \mathcal{Q}^i \mid h_{s-1} = h\},$$

to be the number of trials remaining in epoch i after the last switch from policy h. Additionally, for all $i \in [E+1]$ and $h \in \mathcal{P}^i$ we define

$$a(h,i) := \max\{i' \in [i-1] \cup \{0\} \mid h \in \mathcal{P}^{i'}\},$$

to be the index of the most recent epoch before e^i in which h was relevant. We then define $b(h,i) := e^i - e^{(a(h,i)+1)}$.

For now let us assume, without loss of generality, that an epoch starts on a switch. That is, for all $i \in [E]$ we have $e^i \in \mathcal{Z}$. Let us fix some $h \in \mathcal{H}$ and $i \in [E]$ with $h \in \mathcal{P}^i$. Let $t' := \min\{s \in \mathcal{Z}^i \mid h_s = h\}$ denote the first trial that the policy h occurs in epoch i. We have two possible cases: either $h \notin \Psi^i$ (meaning $h \in \mathcal{P}^{i-1}$, the pool of our previous epoch) or $h \in \Psi^i$ (meaning $h \notin \mathcal{P}^{i-1}$ and h has therefore entered our pool in this epoch).

Case 1: $(h \notin \Psi^i)$ This implies that $h \in \mathcal{P}^{i-1}$. In this case we have

$$\tilde{d}_i^{t'} + \hat{d}_{a(h,i)}(h) = \tilde{d}_i^{t'} + \hat{d}_{(i-1)}(h) \geq \tilde{g}^{t'},$$

and since both terms on the left-hand side are greater than or equal to one, we therefore have

$$\ln(e\tilde{g}^{t'}) \leq \ln(e\tilde{d}_i^{t'}) + \ln(e\hat{d}_{a(h,i)}(h)). \tag{B.21}$$

Case 2: $(h \in \Psi^i)$ This implies that $h \notin \mathcal{P}^{i-1}$. In this case we have

$$\tilde{d}_i^{t'} + b(h,i) + \hat{d}_{a(h,i)}(h) \geq \tilde{g}^{t'} ,$$

and since both terms on the left-hand side are greater than or equal to one, we therefore have

$$\ln\left(e\tilde{g}^{t'}\right) \leq \ln\left(e\tilde{d}_i^{t'}\right) + \ln\left(eb(h,i)\right) + \ln\left(e\hat{d}_{a(h,i)}(h)\right). \tag{B.22}$$

For all $t \in \mathcal{Z}^i \setminus \{t'\}$ such that $h_t = h$ we have $\tilde{d}_i^t = \tilde{g}^t$, and thus at these switching points we have

$$\ln\left(e\tilde{g}^t\right) = \ln\left(e\tilde{d}_i^t\right). \tag{B.23}$$

Now let

$$\tilde{n}_i(h) := \sum_{t \in \mathcal{Z}^i : h_t = h} \ln\left(e\tilde{d}_i^t\right).$$

From (B.21), (B.22), and (B.23) we then have

$$\sum_{t \in \mathcal{Z}^i : h_t = h} \ln\left(e\tilde{g}^t\right) = \tilde{n}_i(h) + \ln\left(e\hat{d}_{a(h,i)}(h)\right) + [\![h \in \Psi^i]\!] \ln\left(eb(h,i)\right). \tag{B.24}$$

Note that if $h \notin \Psi^{E+1}$, then (since this implies $h \in \mathcal{P}^E$) we have $\hat{g}(h) = \hat{d}_E(h)$, and if $h \in \Psi^{E+1}$ then $\hat{g}(h) = b(h, E+1) + \hat{d}_{a(h,E+1)}(h)$. Thus in either case we have

$$\ln\left(e\hat{g}(h)\right) = \ln\left(e\hat{d}_{a(h,E+1)}(h)\right) + [\![h \in \Psi^{E+1}]\!] \ln\left(eb(h,(E+1))\right). \tag{B.25}$$

Now let

$$\hat{n}(h) := \sum_{i \in [E+1] : h \in \mathcal{P}^i} [\![h \in \Psi^i]\!] \ln\left(eb(h,i)\right)$$

Summing (B.24) over all $i \in [E]$ with $h \in \mathcal{P}^i$ and adding (B.25) gives

$$\ln\left(e\hat{g}(h)\right) + \sum_{t \in \mathcal{Z} : h_t = h} \ln\left(e\tilde{g}^t\right) = \sum_{i \in [E] : h \in \mathcal{P}^i} \left(\tilde{n}_i(h) + \ln\left(e\hat{d}_i(h)\right)\right) + \hat{n}(h). \tag{B.26}$$

Finally let

$$n_i(h) := \sum_{t \in \mathcal{Z}^i : h_t = h} \ln(eg^t).$$

Since the start of each epoch coincides with a switch we have, for all $i \in [E]$ and $t \in \mathcal{Z}^i$ with $h_t = h$ that $d_i^t = g^t$, thus $\ln(eg^t) = \ln(ed_i^t)$, and

$$\sum_{t \in \mathcal{Z} : h_t = h} \ln(eg^t) = \sum_{i \in [E] : h \in \mathcal{P}^i} n_i(h).$$

Combining this with (B.26) and summing over all $h \in \mathcal{P}$ gives us, by (B.20) that

$$\begin{aligned}
\bar{C} &= M \ln \frac{N}{M} + \sum_{h \in \mathcal{P}} -\ln\left(f(\boldsymbol{c}^h)\right) \\
&\leq M \ln \frac{N}{M} + 2 \sum_{t \in \mathcal{Z}} \left(\ln(eg^t) + \ln(e\tilde{g}^t)\right) + 2 \sum_{h \in \mathcal{P}} \ln(e\hat{g}(h)) \\
&\leq M \ln \frac{N}{M} + 2 \sum_{i \in [E]} \sum_{h \in \mathcal{P}^i} \left(n_i(h) + \tilde{n}_i(h) + \ln(e\hat{d}_i(h))\right) + 2 \sum_{h \in \mathcal{P}} \hat{n}(h). \quad \text{(B.27)}
\end{aligned}$$

We now bound the right-hand side of (B.27) using Jensen's inequality and the concavity of the logarithm function, such that for a given $k \in \mathbb{N}$, and a set $\{u_i \mid i \in [k]\} \subseteq \mathbb{N}$ we have

$$\sum_{i \in [k]} \ln(eu_i) \leq k \ln\left(\frac{e \sum_{i \in [k]} u_i}{k}\right). \quad \text{(B.28)}$$

First let us fix some epoch $i \in [E]$. We note first that for all $h \in \mathcal{P}^i$ we have

$$\hat{d}_i(h) + \sum_{t \in \mathcal{Z}^i : h_t = h} (d_i^t + \tilde{d}_i^t) = T^i,$$

so summing over $h \in \mathcal{P}^i$ gives

$$\sum_{h \in \mathcal{P}^i} \hat{d}_i(h) + \sum_{t \in \mathcal{Z}^i} (d_i^t + \tilde{d}_i^t) = M^i T^i.$$

Noting also that

$$\sum_{t\in\mathcal{Z}^i} d_i^t = T^i\,, \tag{B.29}$$

we then have

$$\sum_{h\in\mathcal{P}^i} \hat{d}_i(h) + \sum_{t\in\mathcal{Z}^i} \tilde{d}_i^t = (M^i - 1)T^i\,. \tag{B.30}$$

The left-hand side of (B.29) has K^i terms that are summed and the left-hand side of (B.30) has $M^i + K^i$ terms that are summed. Thus combining these with (B.28) gives

$$\sum_{t\in\mathcal{Z}^i} \ln\left(e d_i^t\right) \leq K^i \ln\left(\frac{eT^i}{K^i}\right), \tag{B.31}$$

and

$$\sum_{h\in\mathcal{P}^i} \ln\left(e\hat{d}_i(h)\right) + \sum_{t\in\mathcal{Z}^i} \ln\left(e\tilde{d}_i^t\right) \leq (M^i + K^i) \ln\left(\frac{eM^iT^i}{M^i + K^i}\right), \tag{B.32}$$

respectively. Summing (B.31) and (B.32) then gives

$$\sum_{h\in\mathcal{P}^i} \left(n_i(h) + \tilde{n}_i(h) + \ln\left(e\hat{d}_i(h)\right)\right) \leq K^i \ln\left(\frac{eT^i}{K^i}\right) + (M^i + K^i) \ln\left(\frac{eM^iT^i}{M^i + K^i}\right).$$

Since $M^i \leq K^i$ this implies

$$\sum_{h\in\mathcal{P}^i} \left(n_i(h) + \tilde{n}_i(h) + \ln\left(e\hat{d}_i(h)\right)\right) \in \mathcal{O}\left(K^i \ln\left(\frac{M^iT^i}{K^i}\right)\right). \tag{B.33}$$

All that remains is to bound $\sum_{h\in\mathcal{P}} \hat{n}(h)$. Note first that for all $h \in \mathcal{P}$ we have

$$\sum_{i\in[E+1]:h\in\mathcal{P}^i} [\![h \in \Psi^i]\!] b(h,i) \leq T\,,$$

and thus

$$\sum_{h\in\mathcal{P}} \sum_{i\in[E+1]:h\in\mathcal{P}^i} [\![h \in \Psi^i]\!] b(h,i) \leq MT\,.$$

Since for all $h \in \mathcal{P}$, the above sum does not include h if $h \in \mathcal{P}^i$ for all $i \in [E]$,

the number of terms summed in the left-hand side of this equation is in $\mathcal{O}(\Phi)$, and hence by (B.28) again we have

$$\sum_{h\in\mathcal{P}} \hat{n}(h) = \sum_{h\in\mathcal{P}} \sum_{i\in[E+1]:h\in\mathcal{P}^i} [\![h \in \Psi^i]\!] \ln\left(eb(h,i)\right) \in \mathcal{O}\left(\Phi \ln\left(\frac{MT}{\Phi}\right)\right) . \quad \text{(B.34)}$$

Substituting (B.33) and (B.34) into (B.27) gives

$$\bar{C} \in \mathcal{O}\left(M \ln \frac{N}{M} + \sum_{i\in E^i} K^i \ln\left(\frac{M^i T^i}{K^i}\right) + \Phi \ln\left(\frac{MT}{\Phi}\right)\right) ,$$

which completes the proof. □

Appendix C

A Slow Implementation of AdaptLTM

In this section we give an alternative implementation of AdaptLTM, with a slightly worse computational per-trial time complexity of $\mathcal{O}(N \log t)$-time on trial t.

This implementation does not store $\pi_h^t = \prod_{s=1}^{t-1} \psi_h^s$, in order to compute

$$\sigma_h^{t+1} = \psi_h^t \sigma_h^t - \pi_h^{t+1} \xi_h^t + \tilde{\xi}_h^t$$

and

$$\tilde{\sigma}_h^{t+1} = \tilde{\sigma}_h^t - \tilde{\xi}_h^t + \pi_h^{t+1} \xi_h^t$$

on each trial. Instead, on each trial we require a sum over $\beta_{q,h}^t$ for all $q \in \mathbb{N} \cup \{0\}$, and $\beta_{q,h}^t$ is updated for all $q \in \mathbb{N} \cup \{0\}$. Since $\beta_{q,h}^t$ is only non-zero for $q \leq \max_{s \in [t]} q_s + 1$. This takes $\mathcal{O}(\log t)$ time on trial t per policy.

Routine 13 initialize(h)

1: **for** $q \in \mathbb{N} \cup \{0\}$ **do**
2: $\quad \beta^1_{q,h} := 0$
3: $\quad \tilde{\beta}^1_{q,h} := 0$
4: $\quad i^1_q := 0$
5: **end for**
6: $\beta^1_{0,h} \leftarrow \alpha$
7: $\tilde{\beta}^1_{1,h} \leftarrow \frac{1-\alpha}{2}$

Routine 14 get(t, h)

1: **return** $\left(\sum_{q \in \mathbb{N} \cup \{0\}} \left(2^q - \alpha i^t_q\right) \beta^t_{q,h}, \sum_{q \in \mathbb{N} \cup \{0\}} \left(2^q - \alpha i^t_q\right) \tilde{\beta}^t_{q,h} \right)$

Routine 15 update(t, h)

1: $q_t \leftarrow \text{tree-height}(t)$
2: $\xi^t_h \leftarrow \alpha \psi^t_h \sum_{q \in \mathbb{N} \cup \{0\}} \beta^t_{q,h}$
3: $\tilde{\xi}^t_h \leftarrow \alpha \sum_{q \in \mathbb{N} \cup \{0\}} \tilde{\beta}^t_{q,h}$
4: **if** $q_t = 0$ **then**
5: $\quad \beta^{t+1}_{0,h} \leftarrow \tilde{\xi}^t_h$
6: $\quad \tilde{\beta}^{t+1}_{0,h} \leftarrow \xi^t_h$
7: $\quad \beta^{t+1}_{1,h} \leftarrow (1-\alpha)\psi^t_h \beta^t_{1,h} + \frac{(1-\alpha)}{2} \psi^t_h \beta^t_{0,h}$
8: $\quad \tilde{\beta}^{t+1}_{1,h} \leftarrow (1-\alpha)\tilde{\beta}^t_{1,h} + \frac{(1-\alpha)}{2} \tilde{\beta}^t_{0,h}$
9: **else**
10: $\quad \beta^{t+1}_{0,h} \leftarrow (1-\alpha)\psi^t_h \beta^t_{0,h} + \tilde{\xi}^t_h$
11: $\quad \tilde{\beta}^{t+1}_{0,h} \leftarrow (1-\alpha)\tilde{\beta}^t_{0,h} + \xi^t_h$
12: $\quad \beta^{t+1}_{q_t,h} \leftarrow 0$
13: $\quad \tilde{\beta}^{t+1}_{q_t,h} \leftarrow 0$
14: $\quad \beta^{t+1}_{(q_t+1),h} \leftarrow (1-\alpha)\psi^t_h \beta^t_{(q_t+1),h} + \frac{(2-\alpha)}{4} \psi^t_h \beta^t_{q_t,h}$
15: $\quad \tilde{\beta}^{t+1}_{(q_t+1),h} \leftarrow (1-\alpha)\tilde{\beta}^t_{(q_t+1),h} + \frac{(2-\alpha)}{4} \tilde{\beta}^t_{q_t,h}$
16: **end if**
17: $i^{t+1}_0 \leftarrow 0$
18: $i^{t+1}_{q_t} \leftarrow 0$
19: $i^{t+1}_{q_t+1} \leftarrow 0$
20: **for** $q \in \mathbb{N} \setminus \{q_t, q_t + 1\}$ **do**
21: $\quad \beta^{t+1}_{q,h} \leftarrow \psi^t_h \beta^t_{q,h}$
22: $\quad \tilde{\beta}^{t+1}_{q,h} := \tilde{\beta}^t_{q,h}$
23: $\quad i^{t+1}_q \leftarrow i^t_q + 1$
24: **end for**

Appendix D

Computing the tree-height Function

In this section, we show how the tree-height function can be computed online in $\mathcal{O}(1)$ time per trial. The computation requires a data structure based on the quantities $\{\hat{\pi}_q^t \mid t, q \in \mathbb{N} \cup \{0\}\}$ defined as follows. Given $t, q \in \mathbb{N} \cup \{0\}$, we first define

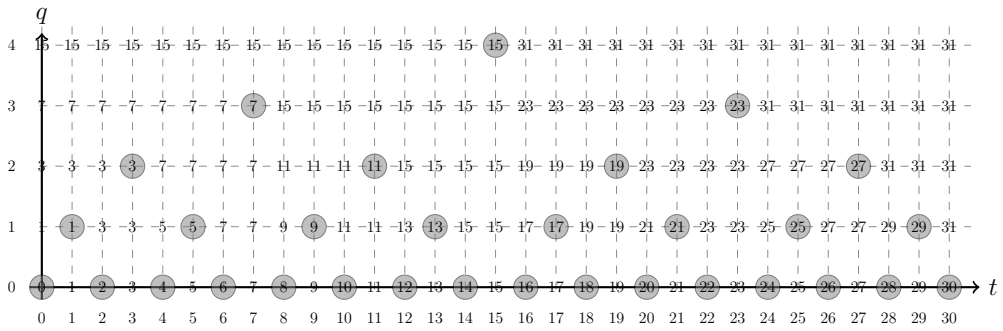
$$\hat{\epsilon}_{t,q} := \min \{s \in \mathbb{N} \cup \{0\} \mid s \geq t \wedge \exists m \in \mathbb{N} : s + 1 = m2^q\},$$

which for a given q is the smallest integer no smaller than t such that $s+1$ is a multiple of 2^q. Intuitively for a given trial t, and level q, $\hat{\epsilon}_{t,q}$ is the closest trial (including or after t) which has a tree-height of at least q (see Figure D.1a). Given some $\hat{\epsilon}_{t,q} \in \mathbb{N} \cup \{0\}$, we then define

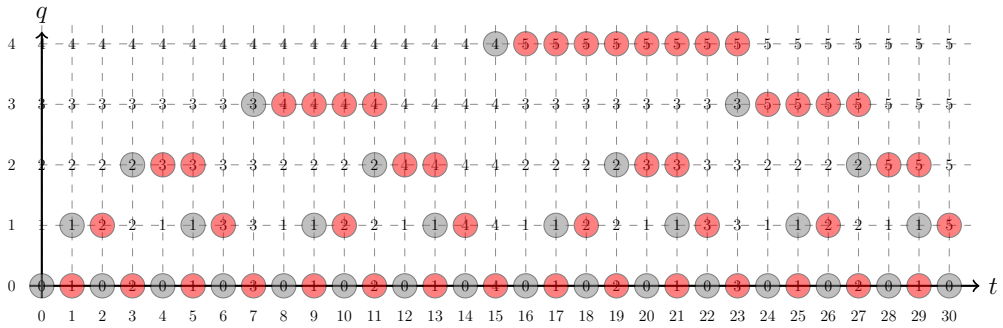
$$\hat{\pi}_q^t := \text{tree-height}(\hat{\epsilon}_{t,q})$$

to be the tree height of that quantity (see Figure D.1b). Note that $\hat{\pi}_q^t$ has the following properties for all $t \in \mathbb{N} \cup \{0\}$:

1. $\hat{\pi}_0^t = q_t$.
2. $\hat{\pi}_{q_t}^{t+1} = \hat{\pi}_{q_t+1}^t$.
3. For all $q < q_t$ we have $\hat{\pi}_q^{t+1} = q$.
4. For all $q > q_t$ we have $\hat{\pi}_q^{t+1} = \hat{\pi}_q^t$.



(a) $\hat{\epsilon}_{t,q}$.



(b) $\hat{\pi}^t_q$.

Figure D.1: (a) At coordinates (t, q) the value of $\hat{\epsilon}_{t,q}$ is shown. (b) At coordinates (t, q) the value of $\hat{\pi}^t_q$ is shown. In gray circles the tree-height function is shown. In red circles are the values of $\hat{\pi}^t_q$ maintained by Algorithm 8 (on each trial).

Our algorithm requires a data structure that stores, on trial t, the set

$$\hat{\omega}^t := \{q \in \mathbb{N} \cup \{0\} \mid (\hat{\pi}^t_q \neq q) \wedge (\hat{\pi}^t_{q'} \leq q\,, \forall q' < q)\}\,,$$

and for all $q \in \hat{\omega}^t$, our data structure stores the value $\hat{\pi}^t_q$. This can be done using a hashmap or a simple look-up table, which we denote $\hat{\pi}^t$ in our algorithm.

Since for all $q \in \mathbb{N} \cup \{0\}$ we have $\hat{\pi}^0_q = q$, the set is initialized with $\hat{\omega}^1 = \{0\}$, and $\hat{\pi}^1$ has $\hat{\pi}^1_0 = 1$ stored. On each trial, the algorithm retrieves q_t which is either 0 on even trials (and we have $0 \notin \hat{\omega}^t$), or we have $q_t = \hat{\pi}^t_0$ (from property 1), which is stored in $\hat{\pi}^t$ when $0 \in \hat{\omega}^t$. After retrieving q_t, our algorithm updates $\hat{\omega}^t$ and $\hat{\pi}^t$ by removing $q_t + 1$ from $\hat{\omega}^t$ (if $q_t + 1 \in \hat{\omega}^t$), and removing the value of $\hat{\pi}^t_{q_t+1}$ from $\hat{\pi}^t$ (if $q_t + 1 \in \hat{\omega}^t$), due to property 3 and the fact that this value is now equal to $\hat{\pi}^t_{q_t}$ (property 2) and is stored in $\hat{\pi}^t$. If $q_t + 1 \notin \hat{\omega}^t$, then we add $\hat{\pi}^t_{q_t}$ to $\hat{\pi}^t$ (which is equal to $q_t + 1$ from properties 2 and 4). See Figure D.1b

Algorithm 8 Computing $q_t = \text{tree-height}(t)$ in $\mathcal{O}(1)$ time

init: $\hat{\omega}^1 \leftarrow \{0\}$; $\hat{\pi}^1 \leftarrow$ hashMap(key, value)
$\hat{\pi}^1$.put$(0, 1)$
for $t = 1, \ldots, T$ **do**
    if $0 \notin \hat{\omega}^t$ **then**
        $q_t \leftarrow 0$
    else
        $q_t \leftarrow \hat{\pi}^t$.get$(0)$
        $\hat{\omega}^t \leftarrow \hat{\omega}^t \setminus \{0\}$
    end if
    if $q_t \notin \hat{\omega}^t$ **then**
        $\hat{\omega}^t \leftarrow \hat{\omega}^t \cup \{q_t\}$
    end if
    if $q_t + 1 \notin \hat{\omega}^t$ **then**
        $\hat{\pi}^t$.put$(q_t, q_t + 1)$
    else
        $\hat{\pi}^t$.put$(q_t, \hat{\pi}^t$.get$(q_t + 1))$
        $\hat{\pi}^t$.remove$(q_t + 1)$
        $\hat{\omega}^t \leftarrow \hat{\omega}^t \setminus \{q_t + 1\}$
    end if
    $\hat{\omega}^{t+1} \leftarrow \hat{\omega}^t$
    $\hat{\pi}^{t+1} \leftarrow \hat{\pi}^t$
end for

for an illustration of the values stored by our algorithm).

Our update for this method, for $t \in \mathbb{N}$, is given in Algorithm 8. This algorithm requires $\mathcal{O}(1)$ time per trial and $\mathcal{O}(\log t)$ space.

Appendix E

Adaptive Long-Term Memory Experiments

In this section we give the values of the parameters used in our experiments in Chapter 5.

Table E.1: Values of parameters of the algorithms included in our experiments. For each algorithm we adopt the notation of the cited paper in naming the parameter.

Algorithm	Parameter	Value
EXP4 [85]	η	0.63
Fixed Share [30]	η	8.90
	α	$\frac{K}{T-1} = \frac{215}{21599}$
PBTS [2]	η	8.07
	$\theta(w\|s)$	$\frac{K}{(M-1)(T-1)} = \frac{215}{23\times 21599}$
	$\theta(s\|w)$	$\frac{K}{T-1} = \frac{215}{21599}$
	$\theta(w)$	$\frac{1}{M} = \frac{1}{24}$
PBTS* [2]	η	11.83
	$\theta(w\|s)$	$\frac{K}{(\tilde{M}-1)(T-1)} = \frac{215}{2\times 21599}$
	$\theta(s\|w)$	$\frac{K}{T-1} = \frac{215}{21599}$
	$\theta(w)$	$\frac{1}{\tilde{M}} = \frac{1}{3}$
Proj FS [37]	η	8.28
	α	$\frac{K}{T-1} = \frac{215}{21599}$
PoDS-θ	η	7.66
	α	$\frac{K}{T-1} = \frac{215}{21599}$
	θ	$\frac{K-M+1}{(M-1)(T-2)} = \frac{192}{23\times 21598}$
AdaptLTM	η	7.62
	α	$\frac{1}{4}$
AdaptLTM*	η	10.62
	α	0.01

Bibliography

[1] Olivier Bousquet and Manfred K Warmuth. Tracking a small set of experts by mixing past posteriors. *Journal of Machine Learning Research*, 3(Nov):363–396, 2002.

[2] Wouter M Koolen, Dmitry Adamskiy, and Manfred K Warmuth. Putting Bayes to sleep. In *NIPS*, pages 135–143, 2012.

[3] Mark Herbster and James Robinson. Online prediction of switching graph labelings with cluster specialists. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[4] James Robinson and Mark Herbster. Improved regret bounds for tracking experts with memory. In *Advances in Neural Information Processing Systems*, volume 34, pages 7625–7636, 2021.

[5] Stephen Pasteris, James Robinson, Massimiliano Pontil, and Mark Herbster. Adaptive long-term memory for experts and bandits. Submitted for review, 2023.

[6] N. Littlestone. Learning Quickly When Irrelevant Attributes Abound: A New Linear-Threshold Algorithm. *Machine Learning*, 2(4):285–318, 1988.

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

[8] V. Vovk. Aggregating strategies. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, COLT '90, pages 371–386, 1990.

[9] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

[10] 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, 1997.

[11] Dmitry Adamskiy, Wouter M Koolen, Alexey Chernov, and Vladimir Vovk. A closer look at adaptive regret. *The Journal of Machine Learning Research*, 17(1):706–726, 2016.

[12] Mark Herbster, Stephen Pasteris, and Lisa Tse. Online multitask learning with long-term memory. In *Advances in Neural Information Processing Systems*, volume 33, pages 17779–17791, 2020.

[13] B. S. Kerner. Experimental features of self-organization in traffic flow. *Phys. Rev. Lett.*, 81:3797–3800, 1998.

[14] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[15] D. J. Klein and M. Randić. Resistance distance. *Journal of mathematical chemistry*, 12(1):81–95, 1993.

[16] M. Herbster, G. Lever, and M. Pontil. Online prediction on large diameter graphs. In *Proceedings of the 21st International Conference on Neural Information Processing Systems*, NIPS '08, pages 649–656, 2008.

[17] M. Herbster and G. Lever. Predicting the labelling of a graph via minimum p-seminorm interpolation. In *COLT 2009 - The 22nd Conference on Learning Theory*, 2009.

[18] N. Cesa-Bianchi, C. Gentile, F. Vitale, and G. Zappella. Random spanning trees and the prediction of weighted graphs. *Journal of Machine Learning Research*, 14(1):1251–1284, 2013.

[19] F. Vitale, N. Cesa-Bianchi, C. Gentile, and G. Zappella. See the tree through the lines: The shazoo algorithm. In *Advances in Neural Information Processing Systems 23*, pages 1584–1592, 2011.

[20] M. Herbster, S. Pasteris, and S. Ghosh. Online prediction at the limit of zero temperature. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 2935–2943, 2015.

[21] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 19–26, 2001.

[22] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML '03, pages 912–919, 2003.

[23] M. Belkin and P. Niyogi. Semi-supervised learning on riemannian manifolds. *Machine learning*, 56(1-3):209–239, 2004.

[24] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS '03, pages 321–328, 2003.

[25] M. Herbster, M. Pontil, and L. Wainer. Online learning over graphs. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pages 305–312, 2005.

[26] N. Cesa-Bianchi, C. Gentile, and F. Vitale. Fast and optimal prediction on a labeled tree. In *Proceedings of the 22nd Annual Conference on Learning Theory*, pages 145–156. Omnipress, 2009.

[27] M. Herbster, S. Pasteris, and M. Pontil. Predicting a switching sequence of graph labelings. *Journal of Machine Learning Research*, 16(1):2003–2022, 2015.

[28] A. Rakhlin and K. Sridharan. Efficient online multiclass prediction on graphs via surrogate losses. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, pages 1403–1411, 2017.

[29] M. Herbster and M. Pontil. Prediction on a graph with a perceptron. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, pages 577–584, 2006.

[30] Mark Herbster and Manfred K Warmuth. Tracking the best expert. *Machine learning*, 32(2):151–178, 1998.

[31] Vladimir Vovk. Derandomizing stochastic prediction strategies. *Machine Learning*, 35(3):247–282, 1999.

[32] András György, Tamás Linder, and Gábor Lugosi. Tracking the best of many experts. In *International Conference on Computational Learning Theory*, pages 204–216. Springer, 2005.

[33] W. M. Koolen and S. Rooij. Combining expert advice efficiently. In *21st Annual Conference on Learning Theory - COLT 2008*, pages 275–286, 2008.

[34] N Cesa-Bianchi, P Gaillard, G Lugosi, and G Stoltz. Mirror descent meets fixed share (and feels no regret). In *Conference on Neural Information Processing Systems*, volume 2, pages 989–997, 2012.

[35] E. Hazan and C. Seshadhri. Adaptive algorithms for online decision problems. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(088), 2007.

[36] Amit Daniely, Alon Gonen, and Shai Shalev-Shwartz. Strongly adaptive online learning. In *International Conference on Machine Learning*, pages 1405–1411. PMLR, 2015.

[37] Mark Herbster and Manfred K Warmuth. Tracking the best linear predictor. *Journal of Machine Learning Research*, 1(Sep):281–309, 2001.

[38] J. Kivinen, A.J. Smola, and R.C. Williamson. Online learning with kernels. *Trans. Sig. Proc.*, 52(8):2165–2176, 2004.

[39] Kwang-Sung Jun, Francesco Orabona, Stephen Wright, and Rebecca Willett. Improved strongly adaptive online learning using coin betting. In *Artificial Intelligence and Statistics*, pages 943–951. PMLR, 2017.

[40] Yuri Kalnishkan, Dmitry Adamskiy, Alexey Chernov, and Tim Scarfe. Specialist experts for prediction with side information. In *2015 IEEE international conference on data mining workshop (ICDMW)*, pages 1470–1477. IEEE, 2015.

[41] Yoav Freund. Private Communication, 2000. Also posted on `http://www.learning-theory.org/`.

[42] R. Lyons and Y. Peres. *Probability on Trees and Networks*. Cambridge University Press, New York, NY, USA, 1st edition, 2017.

[43] D. B. Wilson. Generating random spanning trees more quickly than the cover time. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 296–303, 1996.

[44] O. H. M. Padilla, J. Sharpnack, J. G. Scott, and R. J. Tibshirani. The dfs fused lasso: Linear-time denoising over general graphs. *Journal of Machine Learning Research*, 18(1):1–36, 2018.

[45] Joel Veness, Martha White, Michael Bowling, and András György. Partition tree weighting. In *Data Compression Conference*, pages 321–330. IEEE, 2013.

[46] Marie Devaine, Pierre Gaillard, Yannig Goude, and Gilles Stoltz. Forecasting electricity consumption by aggregating specialized experts. *Machine Learning*, 90(2):231–260, 2013.

[47] M. Herbster. Tracking the best expert II. Unpublished manuscript, 1997.

[48] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[49] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[50] Volodimir G Vovk. Aggregating strategies. *Proc. of Computational Learning Theory, 1990*, 1990.

[51] David Haussler, Jyrki Kivinen, and Manfred K Warmuth. Sequential prediction of individual sequences under general loss functions. *IEEE Transactions on Information Theory*, 44(5):1906–1925, 1998.

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

[53] Dirk Hoeven, Tim Erven, and Wojciech Kotłowski. The many faces of exponential weights in online learning. In *Conference On Learning Theory*, pages 2067–2092. PMLR, 2018.

[54] Robert B Gramacy, Manfred KK Warmuth, Scott Brandt, and Ismail Ari. Adaptive caching by refetching. *Advances in Neural Information Processing Systems*, 15:1489–1496, 2002.

[55] Hai Thanh Nguyen and Katrin Franke. Adaptive intrusion detection system via online machine learning. In *2012 12th International Conference on Hybrid Intelligent Systems (HIS)*, pages 271–277. IEEE, 2012.

[56] Mark Herbster, Stephen Pasteris, and Massimiliano Pontil. Predicting a switching sequence of graph labelings. *J. Mach. Learn. Res.*, 16:2003–2022, 2015.

[57] Kai Zheng, Haipeng Luo, Ilias Diakonikolas, and Liwei Wang. Equipping experts/bandits with long-term memory. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[58] Bin Li and Steven CH Hoi. Online portfolio selection: A survey. *ACM Computing Surveys (CSUR)*, 46(3):1–36, 2014.

[59] Yoram Singer. Switching portfolios. *International Journal of Neural Systems*, 8(04):445–455, 1997.

[60] Avrim Blum and Adam Kalai. Universal portfolios with and without transaction costs. *Machine Learning*, 35(3):193–205, 1999.

[61] Puja Das, Nicholas Johnson, and Arindam Banerjee. Online lazy updates for portfolio selection with transaction costs. In *AAAI*. Citeseer, 2013.

[62] Suleyman S Kozat and Andrew C Singer. Universal switching portfolios under transaction costs. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5404–5407. IEEE, 2008.

[63] Bin Li, Jialei Wang, Dingjiang Huang, and Steven CH Hoi. Transaction cost optimization for online portfolio selection. *Quantitative Finance*, 18(8):1411–1424, 2018.

[64] Najim Al-Baghdadi, David Lindsay, Yuri Kalnishkan, and Sian Lindsay. Practical investment with the long-short game. In *Conformal and Probabilistic Prediction and Applications*, pages 209–228. PMLR, 2020.

[65] David P Helmbold, Robert E Schapire, Yoram Singer, and Manfred K Warmuth. On-line portfolio selection using multiplicative updates. *Mathematical Finance*, 8(4):325–347, 1998.

[66] Volodya Vovk and Chris Watkins. Universal portfolio selection. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 12–23, 1998.

[67] N Littlestone and MK Warmuth. The weighted majority algorithm. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 256–261, 1989.

[68] Yurii Mikhailovich Shtarkov. Switching discrete sources and its universal encoding. *Probl. Inform. Transm.*, 28(3):95–111, 1992.

[69] András Gyorgy, Tamás Linder, and Gábor Lugosi. Efficient tracking of large classes of experts. *IEEE Transactions on Information Theory*, 58(11):6709–6725, 2012.

[70] Wouter M. Koolen and Tim van Erven. Freezing and sleeping: Tracking experts that learn by evolving past posteriors. *CoRR*, abs/1008.4654, 2010.

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

[72] Dravyansh Sharma, Maria-Florina Balcan, and Travis Dick. Learning piecewise lipschitz functions in changing environments. In *International Conference on Artificial Intelligence and Statistics*, pages 3567–3577. PMLR, 2020.

[73] Elad Hazan and Comandur Seshadhri. Efficient learning algorithms for changing environments. In *Proceedings of the 26th annual international conference on machine learning*, pages 393–400, 2009.

[74] Wouter M Koolen and Steven de Rooij. Universal codes from switching strategies. *IEEE Transactions on Information Theory*, 59(11):7168–7185, 2013.

[75] Neri Merhav and Meir Feder. Universal prediction. *IEEE Transactions on Information Theory*, 44(6):2124–2147, 1998.

[76] Qun Xie and Andrew R Barron. Asymptotic minimax regret for data compression, gambling, and prediction. *IEEE Transactions on Information Theory*, 46(2):431–445, 2000.

[77] Gil I Shamir. On asymptotically optimal sequential lossless coding for memoryless switching sources. In *Proceedings IEEE International Symposium on Information Theory,*, page 124. IEEE, 2002.

[78] Gil I Shamir and Daniel J Costello. Universal lossless coding for sources with repeating statistics. *IEEE transactions on information theory*, 50(8):1620–1635, 2004.

[79] Lev M Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217, 1967.

[80] Walid Krichene, Syrine Krichene, and Alexandre Bayen. Efficient bregman projections onto the simplex. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 3291–3298. IEEE, 2015.

[81] Manuel Blum, Robert W. Floyd, Vaughan R. Pratt, Ronald L. Rivest, and Robert Endre Tarjan. Time bounds for selection. *J. Comput. Syst. Sci.*, 7(4):448–461, 1973.

[82] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[83] Robert W Floyd and Ronald L Rivest. Expected time bounds for selection. *Communications of the ACM*, 18(3):165–172, 1975.

[84] Aleksandrs Slivkins. Introduction to multi-armed bandits. *Foundations and Trends in Machine Learning*, 12(1-2):1–286, 2019.

[85] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.

[86] Mark Herbster, Stephen Pasteris, Fabio Vitale, and Massimiliano Pontil. A gang of adversarial bandits. *Advances in Neural Information Processing Systems*, 34:2265–2279, 2021.

[87] Gergely Neu and Julia Olkhovskaya. Efficient and robust algorithms for adversarial linear contextual bandits. In *Conference on Learning Theory*, pages 3049–3068. PMLR, 2020.

[88] Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert Schapire. Taming the monster: A fast and simple algorithm for contextual bandits. In *International Conference on Machine Learning*, pages 1638–1646. PMLR, 2014.

[89] Vasilis Syrgkanis, Akshay Krishnamurthy, and Robert Schapire. Efficient algorithms for adversarial contextual learning. In *International Conference on Machine Learning*, pages 2159–2168. PMLR, 2016.

[90] Haipeng Luo, Chen-Yu Wei, Alekh Agarwal, and John Langford. Efficient contextual bandits in non-stationary worlds. In *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 1739–1776, 06–09 Jul 2018.

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