Combining Strategies Efficiently

High-Quality Decisions from Conflicting Advice

Wouter M. Koolen

Combining Strategies Efficiently

High-Quality Decisions from Conflicting Advice

ILLC Dissertation Series DS-2011-01

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

For further information about ILLC-publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Science Park 904
1098 XH Amsterdam
phone: +31-20-525 6051
fax: +31-20-525 5206
e-mail: illc@uva.nl
homepage: http://www.illc.uva.nl

Combining Strategies Efficiently
High-Quality Decisions from Conflicting Advice

Academisch Proefschrift

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. D.C. van den Boom
ten overstaan van een door het college voor promoties
ingestelde commissie,
in het openbaar te verdedigen in de Agnietenkapel
op donderdag 27 januari 2011, te 14:00 uur

door

Wouter Michiel Koolen

geboren te Groningen

Promotiecommissie

Promotores:

prof. dr. ir. P. M. B. Vitányi
prof. dr. P. D. Grünwald

Overige leden:

prof. dr. P. W. Adriaans
prof. dr. ir. R. Scha
dr. P. J. C. Spreij
dr. ir. T. J. Tjalkens
dr. L. Torenvliet
prof. dr. V. Vovk
prof. dr. M. K. Warmuth

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

Cover design by aÞart, `http://www.apbart.nl`. Based on the fresco *The School of Athens* by Raphael, 1510-1511.

Printed and bound by Ipskamp Drukkers.

ISBN: 978–90–5776–221–5

Sponsors
The investigations were performed at the Centrum Wiskunde en Informatica (CWI).

The research took place within project AFM2.2 of the research program Basic Research in Informatics for Creating the Knowledge Society (BRICKS), subsidised by Besluit Subsidies Investeringen Kennisinfrastructuur (BSIK).

The work was supported by Pattern Analysis, Statistical Modelling and Computational Learning 2 (PASCAL2), a Network of Excellence funded by European Union grant IST-2007-216886.

The author's visits to the University of California, Santa Cruz (UCSC) were supported by NSF grant IIS-0917397.

The author's visit to the Statistical Laboratory at the University of Cambridge was supported by their Department of Pure Mathematics and Mathematical Sciences.

Origin of the Material

This dissertation is based on the following papers:

- Chapter 3 is a significant expansion of technical report

  W. M. Koolen and S. de Rooij. Combining expert advice efficiently. *Computing Research Repository (CoRR)*, abs/0802.2015, Feb. 2008.

  which is itself an expanded version of the conference article

  W. M. Koolen and S. de Rooij. Combining expert advice efficiently. In R. Servedio and T. Zang, editors, *Proceedings of the 21st Annual Conference on Learning Theory (COLT 2008)*, pages 275–286, June 2008.

- Chapter 4 is based on the technical reports

  W. M. Koolen and T. van Erven. Freezing and sleeping: Tracking experts that learn by evolving past posteriors. *Computing Research Repository (CoRR)*, abs/1008.4654, Feb. 2009.

  and

  W. M. Koolen and T. van Erven. Switching between hidden Markov models using Fixed Share. *Computing Research Repository (CoRR)*, abs/1008.4532, Feb. 2010.

- Chapter 5 was published as

  W. M. Koolen and S. de Rooij. Switching investments. In M. Hutter, F. Stephan, V. Vovk, and T. Zeugman, editors, *Proceedings of the 21st International Conference on Algorithmic Learning Theory (ALT 2010)*, LNAI 6331, pages 239–254. Springer, Heidelberg, Oct. 2010.

- Chapter 6 appeared as

  W. M. Koolen, M. K. Warmuth, and J. Kivinen. Hedging structured concepts. In *Proceedings of the 23rd Annual Conference on Learning Theory (COLT 2010)*, pages 93–105, June 2010.

Contents

List of Floats

Figures

Tables

Protocols

Algorithms

Acknowledgments

Many people were involved in the genesis and execution of this dissertation, academically or otherwise. Since this dissertation is about games, it seems suitable to categorise my thanks according to the game in which they are due.

The Graduation Game I am indebted to my promotores Paul Vitányi and Peter Grünwald for their guidance, inspiration and support, and most of all for the incredible amount of freedom they gave me, trusting that beauty arises by rigorously pursuing your own interests.

The Publishing Game I could not have written this dissertation without my co-authors Steven de Rooij, Tim van Erven, Paul Vitányi, Edgar Daylight (a.k.a. Karel van Oudheusden), Martin Ziegler, Manfred Warmuth, Jyrki Kivinen, Leen Stougie, Steven Kelk, Peter van der Gulik, and Harry Buhrmann. It was a pleasure writing with you, and I hope I'll have the opportunity to cross pens with you once more.

Symbol Manipulation Games This work was inspired by my past and present colleagues of INS4/PNA6 Peter Grünwald, Paul Vitányi, Tim van Erven, Steven de Rooij, Wojciech Kotłowski, Alfonso Martinez, Łukasz Dębowski, Nisheeth Srivastava, Thijs van Ommen and Peter Harremoës. You provided a tantalising and vibrant environment for doing research. You taught, guided and encouraged me. You challenged me, and we sparred, debated and competed. And you valued and recognised my accomplishments. You formed me. I can only hope that I returned the favour.

Physical Games Theory needs to be balanced by practise. Vigorous physical practice, that is. I would like to thank the following sparring partners for keeping me sane: Steven de Rooij (squash), Claas-Willem Visser (climbing and hiking), Sirée Koolen-Wijkstra (fencing, cycling and hiking). Thank you: I needed it, and I greatly enjoyed it!

Social Games To stimulate creativity, one needs to tickle the fantasy. Systematically. I would like to thank the following role players for their fantastic inspiration: The *Rotterdam by Night* and *Albany* chronicles, consisting of Maarten Oosten, Julia Krul, Sirée Koolen-Wijkstra and Vincent van der Goes. And the *Allansia* chronicle, consisting of Csaba, Fedde, Erik, Joseph and Sirée.

Culinary Games I want to thank the ESSLLI cooking club for the delicious dinners, interesting discussions and good company. Thank you Marieke van Erp, Paul Groth, Tikitu de Jager, Olga Grigoriadou and Sirée. And thank you Leonard Hugenholtz and Sandra Klaver and also Mirjam Postma and Bjorn Hondelink.

Curiosity Games What does this button do? Let's push it! Nerdy geeks and geeky nerds that flabbergasted and discombobulated me with simple questions to weird answers and vice versa include Steven de Rooij, Maarten Oosten, Vincent van der Goes and Leonard Hugenholtz.

And last but not least, *hors categorie*, I would like to thank my wife Siré Koolen-Wijkstra for her love, support, encouragement, strategic advice and endurance during the intense phases of the process. Finally, I am grateful to my parents Ad and Maite Koolen-Disler for their love. Chapter 5 was written in their care, in their beautiful mountain refuge in Suze, Drôme, France.

Amsterdam Wouter M. Koolen
October, 2010.

Prelude

I want to build intelligent computer systems. Like every computer programmer, I experience great joy and pride when my creation successfully performs a complex task. After half a century of Moore's law[1] we do have the hardware necessary for intelligent behaviour[2]. However it seems that more complicated tasks require more complicated programs. This problem is actually so severe that a lot of intelligent tasks are completely beyond the abilities of current systems. We simply have no clue how to program the desired behaviour. We've hit the so-called *software bottleneck*.

The solution is relatively simple, at least in principle (it is the one encountered in Nature). Instead of trying to *design* a complete system for the desired behaviour, we build a flexible system that can *learn*, and then *train* it to perform the desired task. The design and analysis of such systems is called *machine learning*.

I particularly like the modular approach of building systems that can be immediately put into production untrained. While executing its task, the system continuously learns by receiving (virtual) reward for desired behaviour and (virtual) punishment for *faux pas*. This feedback allows the system to converge to the desired behaviour quickly. The study of this class of systems is called *online learning*. This thesis is the result of four years of doctoral research in online learning.

[1] G. E. Moore noted that the transistor count of integrated circuits doubled every two years.

[2] For example, in November 2009, IBM Research presented its cortical simulator, which runs on supercomputer hardware, and whose simulations exceed the size of the cat cerebral cortex [7]. Earlier milestones include rat-scale and mouse-scale cortices.

Chapter 1

Introduction

$$L - L^*$$

One of the main tasks in Artificial Intelligence is to construct computer systems that *learn* to be successful in any task assigned to them. Indeed, a goal that is as fascinating as it is difficult! Such an ambitious project can only be realised in stages. This dissertation is such a stage.

1.1 The Stage

This dissertation realises successful learning systems for a practically important and diverse class of tasks, known as the *recurring decision problems with full feedback*. That is, problems in which a decision maker repeatedly chooses actions with uncertainty about the quality of each available action. If you perchance want to solve a problem of this kind, then this book is for you. But even if you do not, its overarching approach may provide guidance in addressing other decision problems, the specific methods developed may be used as modular building blocks in the construction of advanced learning systems and the theoretical results derived may provide insight into the philosophy of learning.

This dissertation describes a series of systems that, without any prior knowledge of the task at hand, incrementally learn to choose high-quality actions, in a rigorously defined sense. Namely, each individual system comes with a specific performance guarantee, that relates the quality of its actions to those that would have been chosen by a "prescient" system, or equivalently, by an external observer with hindsight about the quality of all actions. Moreover, we show that each system is computationally efficient in the sense that the required computation time remains modest even if a large amount of data is processed.

Goal & Outline This chapter gently introduces recurring decision problems with full feedback and existing online learning solutions, motivating concepts, goals and methods by example, and incrementally building up to a high-level overview of the research presented in this dissertation. Although the research chapters are self-contained, this informal introduction explains their common conceptual background.

This introduction is itself structured as follows. First, in Section 1.2 we give a definition of and numerous examples from the class of recurring decision problems with full feedback. The first modular step in solving such problems is identifying or creating expert advice, as explained in Section 1.3. The second modular step is to use online learning methods to combine the available expert advice into high-quality actions, as discussed in Section 1.4. We review two fundamental online learning algorithms in Section 1.5, and discuss their guarantees and efficiency. We summarise the online learning approach in Section 1.7. We then place online learning into context by describing related research in Section 1.8. Subsequently, in Section 1.9, we motivate the extended learning tasks studied in this dissertation and discuss their challenges. Then in Section 1.10, we give a short overview of this dissertation's five main chapters. We conclude in Section 1.11.

1.2 Recurring Decision Problems

This dissertation is about *recurring decision problems with full feedback*. That is, problems in which a decision maker repeatedly chooses actions with uncertainty about the quality of each available action. The following concrete examples illustrate the diversity of such problems, their frequency of occurrence and their practical importance.

- A diabetic regularly needs to inject insulin, the desired amount depending on future sugar intake and exercise level.
- A farmer decides whether to irrigate his fields, risking to waste water when it rains later.
- A banker invests her capital in a portfolio of stocks with payoffs determined by future prices.
- A commuter chooses a route daily, whose duration depends on the severity of traffic jams.
- A hybrid car (equipped with both combustion and electric engines) has to select its power source, with overall efficiency depending on price and availability of refuelling and recharging stations.
- An SMS typing assistant predicts the continuation of the current word, with usefulness proportional to the number of key presses saved.

Note that the first few problems are currently addressed by humans, the final few problems are already completely automated, and the problems in the middle are faced by humans with plenty of machine assistance. These boundaries are constantly shifting as more decision problems become automated or assisted.

Side Information Decisions are usually based on *side information*. For example, diabetics typically measure their current blood glucose level, and integrated hybrid car controllers use the route chosen by the onboard navigator as side information. Trading with insider side information is highly profitable and usually illegal.

Loss Function In each problem, the so-called *loss function* governs the set of available actions and the way their quality is measured: better actions incur less loss. The SMS assistant's mistakes are counted by the *0/1 loss* [25]. The banker's capital growth rate is scored by *Cover's loss* [33], and the related *log loss* is appropriate for data compression [34], probability forecasting [39] and hypothesis testing [15]. *Square loss* is also widely used, e.g. in regression and clustering. Total commute time is measured by the *dot loss*, which scores so-called *structured concepts* [92] like home-to-work routes.

Informal Definition A *recurring decision problem* is a task that proceeds in trials. Each trial the decision maker, or *agent*, receives the side information relevant to the current trial. Then the agent chooses an action from a set of available actions, and subsequently incurs the loss associated to the chosen action by the loss function. In recurring decision problem with *full feedback*, the loss of all possible actions is revealed to the agent at the end of the trial. The agent's *cumulative loss* after T trials is the sum of the losses incurred in trials 1 through T.

Our goal is to build an automated agent that learns to exploit the available side information and feedback to incur small cumulative loss. This is accomplished in two modular steps. The first step is mustering or creating experts to interpret the available side information, as explained in the next section. The second step is using the feedback to sequentially combine the expert advice using online learning methods, as explained in Section 1.4.

1.3 Experts

The technical term *expert* denotes any strategy, agent, method or algorithm, for choosing actions based on the available side information. An expert, either human or machine, thus represents a particular approach to a recurring decision problem. An expert may choose actions based on common sense, rules of thumb, human experience and expertise, scientific theories and statistical models. For example,

- In the diabetic example one expert may prescribe to follow the catalogued average insulin production curve of healthy comparable individuals. The patient's physician may recommend a dose based on the patient's medical record. And as a third source of expert advice we may consult a statistical biological model after extracting dietary and exercise clues from the patient's electronic agenda.
- An SMS typing assistant is usually equipped with a dictionary and a set of morphological rules, both annotated with frequencies of occurrence. We may instantiate one expert based on the Dutch language, one on the English language, and one on so-called Textese, `used 4 gr8er input r8`.

- In the banking example a human expert may read all international news and daily trade reports, and recommend investing all capital in shares of IBM today, then in Coca Cola tomorrow etc. A fully automated expert may recommend the portfolio that is optimal under the assumption that stock prices evolve according to some fixed probability distribution.

We assume throughout that suitable experts are present. It is often possible to find human experts, and it is relatively straightforward to create simple automated experts for any decision problem. Even cutting-edge automated experts, based on problem-tailored modelling and incorporating domain-specific knowledge, can in some cases be acquired.

Evidently, different experts capture different aspects of the decision problem, and we desire to intelligently integrate their conflicting advice into high quality decisions. In other words, we want to construct a single superior master expert that combines the advice of many simpler experts.

Recurrence with full feedback allows us to achieve this goal by *learning*, i.e. using the quality of past advice to adjust the relative importance of each expert's advice in our next decision. Let's see how this is done.

1.4 Online Learning

The study of recurring decision problems in the presence of expert advice is the domain of *online learning* [25], an emerging discipline on the interface between computer science, information theory and statistics.

Online-learning methods have been successfully applied to a diverse range of practical problems. Results include e.g. state-of-the-art data compression software [189, 31], improved statistical methods for hypothesis testing and model selection [178, 162, 40], fast natural language entry assistants [184], competitive stock market predictors [33], improved portable device power managers [78, 64] and highly adaptive caching policies [70].

The goal of online learning theory [25] is to develop for each combination of loss function and set of experts an efficient algorithm that guarantees small *regret*, where regret is defined as the difference between the loss of the algorithm and the loss of the best expert. An algorithm with small regret learns to act like the best expert. This im-

Protocol 1.1 Sequential point forecasting

for trial $t = 1, 2, \ldots$ **do**
Receive a prediction $p_{i,t} \in \{\mathsf{y}, \mathsf{n}\}$ for each expert $i \in \{1, \ldots, n\}$.
Choose a prediction $q_t \in \{\mathsf{y}, \mathsf{n}\}$.
Observe the actual outcome $x_t \in \{\mathsf{y}, \mathsf{n}\}$.
Incur a mistake if $q_t \neq x_t$. Expert i incurs a mistake if $p_{i,t} \neq x_t$.
end for

Regret w.r.t. best expert after T trials: $L - L^*$ mistakes, where

$$L := \sum_{t=1}^{T} \mathbf{1}_{q_t \neq x_t} \qquad \text{and} \qquad L^* := \min_i \sum_{t=1}^{T} \mathbf{1}_{p_{i,t} \neq x_t}.$$

plies that if at least one of the experts that the algorithm has access to incurs small loss, then the algorithm itself suffers small loss as well, and hence, as desired, learns to perform its task well. We now introduce the two fundamental regret guarantees that can be obtained.

1.5 Fundamental Online Learning Problems

Recall our farming example, where a farmer needed to decide whether to irrigate or not, and hence needed to predict the rain. We abbreviate the event that it rains to y, and denote its complement by n. We analyse this online learning problem for two different loss functions: 0/1 loss and log loss. In each case our goal is to minimise our *regret*, i.e. the difference between our cumulative loss L and the cumulative loss L^* of the best expert. The regret formula $L - L^*$ serves as this chapter's logo.

1.5.1 Prediction with 0/1 Loss

Say that we have n weather forecasting experts, e.g. employed by n television channels. Every evening, each expert predicts whether it will rain tomorrow or not by quoting either y or n. We then form our own binary prediction based on this advice. The weather is observed the following day, revealing correct and erroneous predictions, that is, providing full feedback about the loss of all possible predictions. Namely,

on outcome x, prediction p suffers 0/1 loss

$$\mathbf{1}_{p \neq x} \;=\; \begin{cases} 0 & \text{if } p = x, \\ 1 & \text{if } p \neq x. \end{cases}$$

This prediction problem is repeated daily, and our goal is to minimise our regret $L - L^*$, i.e. the difference between our cumulative mistake count L and the cumulative mistake count L^* of the best expert. See Protocol 1.1 for a systematic rendering of the timing of the actions and for the formal definition of L and L^*. We illustrate the setup by example. Say after $T = 5$ days the actual rain sequence was

$$x \;=\; \mathsf{y}, \mathsf{y}, \mathsf{y}, \mathsf{n}, \mathsf{y}$$

while the $n = 3$ experts that we consulted sequentially predicted

$$\begin{aligned} p_1 \;&=\; \mathsf{y}, \mathsf{y}, \mathsf{y}, \mathsf{y}, \mathsf{y} && \text{making 1 mistake,} \\ p_2 \;&=\; \mathsf{y}, \mathsf{n}, \mathsf{y}, \mathsf{n}, \mathsf{y} && \text{making 1 mistake,} \\ p_3 \;&=\; \mathsf{n}, \mathsf{n}, \mathsf{n}, \mathsf{n}, \mathsf{n} && \text{making 4 mistakes,} \end{aligned}$$

and we sequentially combined these predictions into our predictions

$$q \;=\; \mathsf{y}, \mathsf{n}, \mathsf{n}, \mathsf{y}, \mathsf{y} \qquad \text{making 3 mistakes.}$$

Then our cumulative loss is $L = 3$, the cumulative loss of the best expert is $L^* = 1$, so that our regret is $L - L^* = 2$ mistakes.

The Hedge Algorithm The Hedge algorithm [59] is a systematic way to combine point predictions. At a high level, the idea is to give more weight to predictions by experts that have suffered small 0/1 loss in the past. Full details are given in Chapter 6. The Hedge algorithm issues randomised predictions, so that L is a random variable. Applied to n experts, the Hedge algorithm guarantees expected regret

$$\mathbb{E}[L] - L^* \;\leq\; \sqrt{2\, L^* \, \ln(n)} + \ln(n) \quad \text{mistakes.} \tag{1.1}$$

This guarantee holds irrespective of the number of days T and without making any stochastic assumptions about the way outcomes arise; it

Protocol 1.2 Sequential probability forecasting

for trial $t = 1, 2, \ldots$ **do**
  Receive a probability $p_{i,t} \in [0,1]$ of y for each expert $i \in \{1, \ldots, n\}$.
  Choose a probability $q_t \in [0,1]$ of y.
  Observe the actual outcome $x_t \in \{\mathsf{y}, \mathsf{n}\}$.
  Suffer log loss $-\log q_t(x_t)$. Expert i suffers log loss $-\log p_{i,t}(x_t)$.
end for

Regret w.r.t. best expert after T trials: $L - L^*$ bits, where

$$L := \sum_{t=1}^{T} -\log q_t(x_t) \quad \text{and} \quad L^* := \min_i \sum_{t=1}^{T} -\log p_{i,t}(x_t).$$

holds for each individual sequence of outcomes x. It implies that the quality of our predictions is close to that of the best expert. In fact, dividing both sides by the number of days T, we see that the overhead per day tends to zero since $L^* \leq T$. As one would expect, it is harder to approximate the overall best expert if there are many experts (n is large). Apparently, it is also harder to predict like the best expert if all experts are bad (L^* is large). This phenomenon is explained in Chapter 2.

Moreover, the Hedge algorithm runs in time $O(n)$ per day. This is clearly optimal, simply consulting the n experts already takes this many steps.

1.5.2 Prediction with Log Loss

Now suppose that our n weather forecasting experts report a *probability* of rain, as is commonly done e.g. on national television, and that we want to combine these forecasts into our own probability. The loss function commonly used for such probability forecasts is the so-called log loss, which is the information-theoretic measure of surprise measured in bits[1]. More precisely, the *log loss* of predicting outcome y with probability $p \in [0,1]$ (and hence outcome n with probability $1 - p$) on

[1] Log loss originated in coding theory, where code lengths are measured in bits. Probability forecasting with log loss and sequential coding are essentially the same, as shown by the Kraft Inequality and Shannon-Fano coding. [34]

actual outcome $x \in \{\mathsf{y}, \mathsf{n}\}$ is

$$-\log p(x) \;:=\; \begin{cases} -\log(p) & \text{if } x = \mathsf{y}, \\ -\log(1-p) & \text{if } x = \mathsf{n}, \end{cases}$$

where log denotes the logarithm to base 2. As expected, assigning high probability to the actual outcome means small loss and vice versa. Note that again by observing the outcome x we get full feedback about the loss of all possible predictions p.

In contrast to the 0/1 loss used for counting mistakes, the log loss is not bounded. In particular, any p that assigns zero probability to the actual outcome x suffers infinite loss. The log loss setup is displayed as Protocol 1.2.

Recall that in the example of the previous section the actual rain sequence after $T = 5$ days was

$$x \;=\; \mathsf{y}, \mathsf{y}, \mathsf{y}, \mathsf{n}, \mathsf{y}.$$

Suppose that our $n = 3$ experts sequentially quoted rain probability

$$\begin{aligned} p_1 &= 0.6, 0.6, 0.6, 0.6, 0.6 && \text{incurring 4.270 bits,} \\ p_2 &= 0.8, 0.7, 0.6, 0.5, 0.4 && \text{incurring 3.895 bits,} \\ p_3 &= 0, \quad 1, \quad 0, \quad 1, \quad 0 && \text{incurring } \infty \text{ bits,} \end{aligned}$$

and we sequentially combined these predictions into our predictions

$$q \;=\; 0.7, 0.6, 0.6, 0.5, 0.5 \qquad \text{incurring 3.989 bits.}$$

Then our cumulative loss is $L = 3.989$ bits, the cumulative loss of the best expert is $L^* = 3.895$ bits, so that our regret is $L - L^* = 0.093$ bits.

The Bayesian Strategy The Bayesian strategy is a systematic way to combine probabilistic predictions. At a high level, the idea is again to give more weight to predictions by experts that have suffered small log loss in the past. Full details are given in Chapter 3. Applied to n experts, the Bayesian strategy guarantees regret

$$L - L^* \;\leq\; \log(n) \text{ bits.} \tag{1.2}$$

Like Hedge, the Bayesian strategy runs in optimal time $O(n)$ per day.

1.5.3 Comparison

Thus, in both cases there is a computationally efficient strategy that guarantees that the quality of our predictions is close to that of the best expert. This means that we can learn to approximately predict like the best expert. Here, as always, "best" means "best with hindsight" at the time T of evaluation.

These results also allow us to quantitatively compare the two prediction problems. The fact that the point forecasting regret bound (1.1) grows as the square root of the loss of the best expert whereas the probability forecasting regret bound (1.2) is completely independent of the expert losses indicates that the first task is harder than the second.

Finally, note that both cases share the same interesting underlying tradeoff. To see this, consider the loss as a function of the number n of experts that we feed into either algorithm. The loss of the best expert is non-increasing in the number of experts used. But the regrets (or at least both bounds) are increasing in the number of experts used. The best cumulative loss is thus obtained by running these algorithms with a reasonably sized set of reasonable experts, so that both the regret and the loss of the best expert are small.

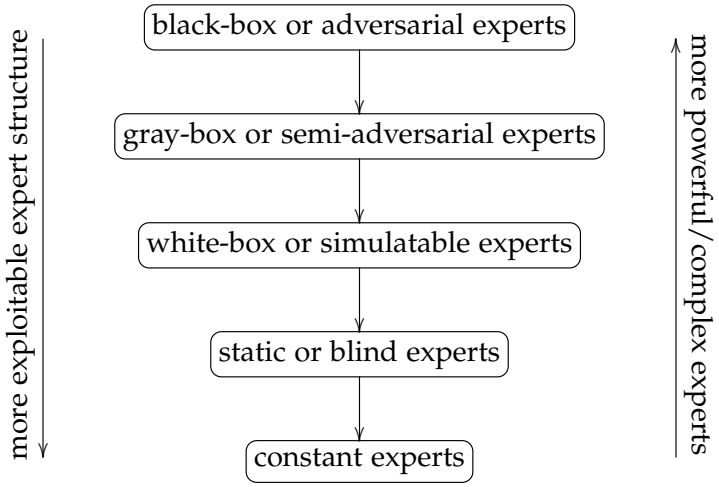
In this section we did not use any background knowledge about the predictions of the experts, and hence obtained regret guarantees that hold irrespective of the expert advice. On the other hand, sometimes explicit knowledge about the experts functioning is available, and may be used to our advantage, to obtain better regret guarantees. In the next section we classify the different types of experts, according to what we know about them.

1.6 Nomenclature and Taxonomy of Experts

We already pointed out in Section 1.3 that the notion of expert is extremely general: an expert is any strategy, agent, method or algorithm, for choosing actions in a given recurring decision problem. In this section we identify different types of experts, and classify them according to how much information we have about their functioning. The resulting hierarchy is displayed in Figure 1.1.

At the bottom we find the simplest *constant experts*. A constant expert always recommends the same action. Next we encounter the

Figure 1.1 Hierarchy of expert types



slightly more general *static* or *blind experts*. A static expert follows a fixed, known sequence of actions, so that its actions only depend on time, but not on the loss of its past actions.

Experts in the subsequent category of *white-box* or *simulatable experts* do adapt their recommendation based on the loss of their past actions, and they do this in a way that is known to us. That is, given past outcomes we can always compute the action they will recommend next. The most general class of experts are the *black-box* or *adversarial experts*. This class includes human experts, which we can consult but which we can not simulate. To obtain performance guarantees for algorithms for decision problems with black-box experts, we have to assume the worst case, that is, we assume that the advice of black-box experts is determined by a malevolent adversary. Finally, the *gray-box* or *semi-adversarial experts* arise when we combine the advice of black-box experts in a simulatable fashion.

Before going into the details of this dissertation, we first summarise what has been presented so far, and discuss related research.

1.7 Summary

In recurring decision problems a decision maker repeatedly chooses actions with uncertainty about the quality of each available action. With full feedback, the quality of each action is revealed after a decision is committed. Online learning advocates the following approach to solving recurring decision problems with full feedback:

1. Identify the set of actions.
2. Identify the loss function.
3. Identify the available side information.
4. Gather or create experts. That is, get access to agents that, each trial, use the side information to recommend an action.
5. Run an algorithm that uses the feedback to combine the expert advice, and that guarantees small regret w.r.t. the best expert.

For step 5 we need algorithms that guarantee small regret w.r.t. the best expert, i.e. that learn to act approximately like the best expert. This dissertation investigates the design and analysis of such algorithms.

Note that the resulting algorithm itself qualifies as an expert, since it sequentially chooses actions. Its output can therefore serve as the expert input in step 4 to an even higher-level algorithm. Such hierarchical combinations of experts can be used to obtain advanced learning algorithms.

1.8 Related Research

Different models of learning are actively researched [126, 154, 8, 95, 106, 127, 179, 35, 74, 160, 17, 150, 25, 71]. There are two main approaches: learning as displaying the correct behaviour vs learning as identifying the underlying rule or process.

1.8.1 Decision-Making

A large class of models for learning agrees with online learning that the objective is to build systems that make good decisions. These models differ in the exact type of problems considered. The closest relative is called *bandit learning*. [176, 155, 158, 50, 24, 55, 10, 118, 143, 5, 26] This

name refers to a gambler who wants to learn which slot machine, or *one-armed bandit*, yields the best overall payoff. In contrast to online learning, the learner only gets *partial feedback*. He observes the loss of his chosen action, but not the loss of the other possible actions. A typical bandit problem is *ad(vertisement) selection*. An ad has to be selected from a large set for presentation to the user, say in the margin of some web page. The system's owner receives some financial reward if the user clicks on the ad. In this setting, the system clearly only observes the user's reaction to the presented ad, and not to all possible ads. Despite the difference, algorithms for the bandit setting often strongly resemble online learning algorithms.

The even more difficult setting called *Reinforcement Learning* arises when the quality of chosen actions is only revealed later, indirectly, or not at all. [1, 9, 12, 13, 14, 20, 11, 22, 36, 30, 46, 48, 45, 42, 44, 49, 52, 37, 47, 54, 56, 57, 61, 60, 67, 62, 69, 63, 68, 76, 82, 83, 88, 89, 90, 96, 98, 94, 93, 97, 103, 109, 105, 110, 86, 107, 131, 114, 122, 116, 134, 133, 113, 121, 123, 120, 132, 124, 130, 119, 111, 115, 117, 135, 125, 137, 136, 138, 145, 141, 142, 139, 140, 144, 148, 152, 149, 151, 157, 156, 153, 147, 167, 159, 165, 169, 168, 164, 166, 175, 170, 171, 174, 183, 191, 187, 188, 51]

In these settings it is often necessary to make assumptions about the environment before anything can be provably learnt.

Methods for making good decisions compared to a class of experts can be found in the literature on different fields. Examples include *universal modelling* in statistics [71], *universal coding* in data compression [34] and *universal portfolios* in finance [33].

1.8.2 Inference

A second set of models for learning pursues a different goal altogether. The crucial difference is that there is no extrapolation into the future. Instead, the system has to create a pattern, rule, description, model, hypothesis or explanation of the situation at hand from examples. Applications include data analysis, data mining, data compression, density estimation, archaeology, clustering and de-noising. Although the goal is exploratory, decision making is sometimes used as a sanity check.

The background so far provided by the introduction allows us to explain the content of this dissertation in slightly more detail.

1.9 Meta-Experts — This Dissertation

This introduction discussed a basic case: learning to behave like the best expert. In the remainder of this dissertation we consider four complex online learning problems, with more ambitious goals. In each case, we start out with a set of "basic" experts, and then create a huge variety of derived "meta"-experts, in such a way that we expect the best meta-expert to be much better than the best basic expert. We consider the following three sets of meta-experts.

- Switching between basic experts. Here a meta-expert switches between basic experts as time progresses. Such meta-experts outperform the best basic expert when the relative quality of the basic experts varies over time. Competing with the best meta-expert means learning when to switch between basic experts.

- Switching between learning basic experts. When basic experts are themselves learning in the sense that their predictions depend on the data that they have observed, we create a more ambitious goal. Namely, each meta-expert switches between basic experts as time progresses, but only lets the currently selected basic expert observe the current outcome. Competing with the best meta-expert means learning how to segment the data, in such a way that for each segment, the best basic expert for that segment can learn to exploit the local patterns well.

- Structured concepts. Consider home-to-work routes. Say that we have for each directed road segment a basic expert that specialises in driving that segment. A meta-expert now is a route from home to work, that uses *all* the basic experts along the path. The loss (commute time) of a meta-expert is the sum of the losses of the basic experts used. Competing with the best meta-expert means learning the shortest path from home to work. Analogous constructions exist for other combinatorial combinations of experts, e.g. sets, spanning trees, permutations etc.

In each case, the meta-experts that we construct are gray-box experts according to the classification of Section 1.6. We may not know how the underlying basic experts function, but we do know exactly how the basic experts are combined.

As before, we then try to build algorithms that have small regret w.r.t. the best meta-expert, i.e. that learn to approximate the best meta-expert. This cannot be achieved by simply applying the fundamental algorithms from Section 1.5 on the set of meta-experts, for two reasons:

- Efficiency. The set of meta-experts is huge, often exponential in the number of basic experts and sometimes even infinite, so that running the algorithm is intractable.
- Performance. The regret bounds (1.1) and (1.2) of the standard algorithms are tight when the experts' actions are independent. But the actions of different meta-experts are highly dependent, since they are derived from the same set of basic experts, so we expect that much better bounds are achievable.

The challenge is hence to exploit the internal structure of the set of meta-experts. That is, we want to design new algorithms that are efficient in terms of the number of basic experts, and that use the dependence between meta-experts to obtain tight regret bounds. This dissertation contains efficient algorithms for each of these classes of meta-experts.

1.10 Organisation of this Dissertation

We now sketch the contents of the main chapters of this dissertation. The next chapter, Chapter 2, is written at an introductory level. It does not assume any prior knowledge about online learning, and contains examples of the concepts used in the remainder. The subsequent four chapters are based on research articles.

1.10.1 Chapter 2: Regret Games (Introductory)

We give a game-theoretic analysis of the simplest online learning problem, the prediction of a sequence of binary outcomes under 0/1 loss with the help of 2 experts. For this simple problem, we compute the minimax, i.e. game-theoretically optimal, regret, and show how to implement the optimal strategy efficiently. We then give special attention to the case that one of the experts is good. We conclude with a new result: we give the optimal algorithm for competing with the set of meta-experts that switch between the 2 basic experts.

1.10.2 Chapter 3: Expert Hidden Markov Models

We show how models for prediction with expert advice can be defined concisely and clearly using hidden Markov models (HMMs); standard algorithms can then be used to efficiently calculate how the expert predictions should be weighted. We focus on algorithms for "tracking the best expert", starting from the Fixed Share algorithm [79, 80], and show how most existing models can be cast as HMMs. We recover the running times and loss bounds for each algorithm, and discuss how they are related. We also describe three new models: (i) models with decreasing switching rate, which run in linear time and for which a fixed switching rate does not have to be specified in advance, (ii) a new generalisation of the fixed share algorithm that is especially well equipped to handle switches that occur in clusters, and (iii) a model tailored to the scenario where the experts have a natural order, and where jumps between them are typically small. This last model is relevant for predicting time series data where parameter drift is expected.

1.10.3 Chapter 4: Freezing & Sleeping

A problem posed by Yoav Freund at the Conference on Learning Theory (COLT) in 2000 is how to efficiently track a small pool of experts out of a much larger set. This problem was solved when Bousquet and Warmuth introduced their mixing past posteriors (MPP) algorithm in 2001.

In Freund's problem the experts would normally be considered black boxes. However, in this chapter we re-examine Freund's problem in case the experts have internal structure that enables them to learn. In this case the problem has two possible interpretations: should the experts learn from all data or only from the subsequence on which they are being followed? The MPP algorithm solves the first case. Our contribution is to generalise MPP to address the second option. The results we obtain apply to any expert structure that can be formalised using (expert) hidden Markov models. Curiously enough, for our interpretation there are *two* natural reference schemes: freezing and sleeping. For each scheme, we provide an efficient prediction strategy and prove the relevant loss bound.

1.10.4 Chapter 5: Switching Investments

We present a simple online two-way trading algorithm that exploits fluctuations in the unit price of an asset. Rather than analysing worst-case performance under some assumptions, we prove a novel, unconditional performance bound that is parameterised either by the actual dynamics of the price of the asset, or by a simplifying model thereof. The algorithm processes T prices in $O(T^2)$ time and $O(T)$ space, but if the employed prior density is exponential, the time requirement reduces to $O(T)$. The result translates to the prediction with expert advice framework, and has applications in data compression and hypothesis testing.

1.10.5 Chapter 6: Hedging Structured Concepts

We develop an online algorithm called *Component Hedge* for learning structured concept classes when the loss of a structured concept sums over its components. Example classes include paths through a graph (composed of edges) and partial permutations (composed of assignments). The algorithm maintains a parameter vector with one non-negative weight per component, which always lies in the convex hull of the structured concept class. The algorithm predicts by decomposing the current parameter vector into a convex combination of concepts and choosing one of those concepts at random. The parameters are updated by first performing a multiplicative update and then projecting back into the convex hull. We show that Component Hedge has optimal regret bounds for a large variety of structured concept classes.

1.11 Conclusion

We introduced the online learning approach to recurring decision problems with full feedback. We first reviewed prediction with expert advice, where the goal is to suffer small regret w.r.t. the best expert. We then discussed more ambitious goals that arise by combining few basic experts into a huge set of meta-experts, and then trying to approximate the best meta-expert. We briefly summarised the sets of meta-experts that are studied in the four research chapters this thesis, and previewed

the results that we obtain in each case, which take the form of algorithms and corresponding regret bounds.

Chapter 2

Regret Games

	y	n
y	$\mathcal{G}_{\mathsf{y}}$	$\mathcal{G}_{\mathsf{n}} + 1$
n	$\mathcal{G}_{\mathsf{y}} + 1$	$\mathcal{G}_{\mathsf{n}}$

2.1 Introduction

The simplest decision problem is the prediction of a binary outcome. For example, we would like to predict whether it is going to rain tomorrow or not, whether a stock's price will be higher tomorrow or lower, etc. Predicting a binary outcome is an important and common task, and it is also a fundamental decision problem because it arises as a special case of many more sophisticated decision problems. We already encountered this prediction problem in Section 1.5.1.

In this chapter we study predicting a binary outcome from the online learning viewpoint, with the goal of understanding both this fundamental decision problem and its solutions. While we present several new results, the chapter also serves an introductory purpose: this chapter requires no prior knowledge about online learning, and illustrates by example the concepts used in the subsequent four chapters. We deliberately keep the setup as simple as possible. That is, we adopt the typical loss function for this scenario, the 0/1 loss, which simply counts the number of mistakes.

We formulate online prediction with experts as a game, and use elementary game theory to solve it. The advantage of this approach is twofold. First, we get the *minimax regret*, a number that expresses the complexity of the binary prediction problem, and gives us a lower bound on the regret of any prediction algorithm. And second, we get the *minimax strategy*, a prediction algorithm that achieves that lower bound, and hence provides the optimal regret guarantee. This game-theoretic approach to online learning was generally believed to be intractable [27]. But two recent breakthrough papers found efficient minimax strategies for prediction [4] and decision-making [6]. These algorithms compete with *black-box* experts, whose predictions and losses are determined completely adversarially.

In this chapter we focus on the simpler *white-box experts* (see the taxonomy in Section 1.6), for which we can compute all future predictions. We first consider the simplest, constant experts. Unexpectedly, by simplifying the prediction problem to its core, we obtain several new results, including a new, optimal, efficient algorithm for the more complex switching experts and for other elaborate classes of white-box experts. We also obtain new insights in existing algorithms by observing how exactly they coincide for our simple setup.

Experts We abstract away the meaning of the outcome, and call the two possible values y and n. We use the two simplest *constant experts*

$$\mathsf{yyyyyyyyyyyyyyyy}\cdots \qquad \text{and} \qquad \mathsf{nnnnnnnnnnnnnnnn}\cdots$$

That is, one expert always predicts y and the other always predicts n. Apart from simplicity, another motivation for using these two experts is the following. If we assume that the outcomes are generated by flipping a biased coin, say loaded towards y, then the expected $0/1$ loss is minimised by *always* predicting y, and vice versa for n. Hence the entire biased coin model is predicted optimally by these two experts.

Note however that the stochastic assumption made in the above argument is merely used as a sanity check to argue that these experts are reasonable, and that we never make any such unverifiable assumptions about the outcome-generating process in any of our prediction problems.

The goal now is to suffer small regret w.r.t. the best expert. In this case, competing with the best constant expert means learning the global trend, i.e. the outcome most frequent in the final sequence. Although seemingly simple, the prediction problems with these two constant experts exhibit many interesting general phenomena, while clearly illustrating the underlying difficulties.

We subsequently consider the more sophisticated *switching experts* that may switch between outcomes, predicting for example

$$\mathsf{yyyyynnnnnnnyyyy}\cdots$$

Whereas competing with the best constant expert allows us to learn the global trend of the sequence of outcomes, competing with switching experts allows us to track local trends. This is useful in applications where we expect the outcome-generating process to sometimes change its characteristics, for example because it can be in several states. Again, the goal is to suffer small regret w.r.t. the best switching expert.

Adversary In each trial of the binary prediction problem, we have uncertainty about the actual outcome. We model this uncertainty by placing the generation of the actual outcome in the hands of a second player, which we call Adversary. The sole purpose of Adversary is to maximally frustrate our learning attempt. Thus we are trying to

minimise our regret while Adversary is trying to maximise it. Our goal is to find a strategy that guarantees small regret against the malevolent Adversary. By preparing for the worst case, we obtain a strategy with regret guarantees that *always* hold. In particular, the validity of our guarantees does not depend on any assumptions about the underlying outcome-generating process.

Goal & Outline In this chapter we model the prediction problem as a mathematical game, stressing the strategic considerations underlying its analysis. We first review elementary strategic game theory in Section 2.2, and then use it to solve predicting a single binary outcome in Section 2.3. We then review the theory of repeated games in Section 2.4, and use it to solve predicting a sequence of binary outcomes in Section 2.5. We apply the solution to related problems in Section 2.6. We then consider the special case that the best expert has small cumulative loss in Section 2.7, and obtain regret bounds that are parametrised accordingly. We demonstrate the generality of the methodology in Section 2.8 and apply it to the set of switching experts in Section 2.9. Section 2.11 concludes by placing these results into context.

2.2 2×2 Strategic Games

In this chapter, we model several prediction problems as mathematical games, and this section introduces the game theory necessary to model a single trial. The game format appropriate for our problems is that of strategic games, in which the players choose their action in parallel. Contemporaneous moves model both our uncertainty about the actual outcome, and our ability to randomise our prediction in a way that Adversary cannot foretell.

In general, a two-player zero-sum 2×2 strategic game is represented by a cost matrix:

		Adversary	
		y	n
We	y	A	B
	n	C	D

We identify with the row player, and call the column player Adversary.

Both players independently choose an action, either y or n. We then incur the cost A, B, C or D associated with the chosen pair of actions. It is our objective to minimise the cost, while Adversary tries to maximise it. In this section we compute our optimal strategy and its guarantee. Our optimal strategy is often a *mixed strategy*, which carefully randomises its action.

Saddle Point We identify a mixed strategy for either player with the probability it assigns to the action y. When we follow strategy $\sigma \in [0,1]$, while Adversary follows strategy $\tau \in [0,1]$, our expected cost is

$$V(\sigma,\tau) \;:=\; \sigma\tau A + \sigma(1-\tau)B + (1-\sigma)\tau C + (1-\sigma)(1-\tau)D.$$

A *minimax* strategy $\hat{\sigma}$ attains $\min_\sigma \max_\tau V(\sigma,\tau)$ while a *maximin* strategy $\hat{\tau}$ attains $\max_\tau \min_\sigma V(\sigma,\tau)$. The celebrated Von Neumann minimax theorem (see [16]) states that a saddle point $\langle\hat{\sigma},\hat{\tau}\rangle$ exists with

$$V(\hat{\sigma},\hat{\tau}) \;=\; \min_\sigma \max_\tau V(\sigma,\tau) \;=\; \max_\tau \min_\sigma V(\sigma,\tau).$$

We call the above quantity the *game value* and denote it by $\mathcal{V}$. The game value $\mathcal{V}$ equals both our *minimax cost* and Adversary's *maximin cost*. In words, our minimax strategy $\hat{\sigma}$ guarantees that our expected cost is at most $\mathcal{V}$, while Adversary's maximin strategy $\hat{\tau}$ guarantees that our expected cost is at least $\mathcal{V}$. Consequently, playing $\hat{\sigma}$ vs $\hat{\tau}$ results in $\mathcal{V}$.

Computing a Saddle Point For 2×2 games it is straightforward to compute the game value $\mathcal{V}$ and construct a saddle point $\langle\hat{\sigma},\hat{\tau}\rangle$, as shown in Table 2.1. Games satisfying any of the first four cases of Table 2.1 have a saddle point in *pure*, i.e. deterministic, strategies. The interesting case is the fifth, where randomisation is essential for both players. A tedious but straightforward case-by-case analysis shows that no pure saddle point exists if and only if

$$(A-B)(C-D) \;<\; 0 \qquad \text{and} \qquad (A-C)(B-D) \;<\; 0. \tag{2.1}$$

We now solve games without pure saddle points. The game-theoretic analyses in all later sections are built upon this workhorse lemma.

Table 2.1 Solution of 2×2 matrix games. The game value $\mathcal{V}$ and saddle point $\langle \hat{\sigma}, \hat{\tau} \rangle$ are shown as a function of the costs A, B, C and D.

$\hat{\sigma}$	$\hat{\tau}$	$\mathcal{V}$	condition
1	1	A	$C \geq A \geq B$
1	0	B	$D \geq B \geq A$
0	1	C	$A \geq C \geq D$
0	0	D	$B \geq D \geq C$
$\frac{D-C}{Z}$	$\frac{D-B}{Z}$	$\frac{AD-BC}{Z}$	condition (2.1)

where $Z = A - B - C + D$

2.2.1. Lemma. *If a game has no saddle point in pure strategies, then the game value $\mathcal{V}$ and unique saddle point $\langle \hat{\sigma}, \hat{\tau} \rangle$ are given by*

$$\mathcal{V} \;=\; \frac{AD - BC}{Z}, \qquad \hat{\sigma} \;=\; \frac{D - C}{Z} \qquad \textit{and} \qquad \hat{\tau} \;=\; \frac{D - B}{Z}$$

where $Z = A - B - C + D$.

Proof. Say that we play action y with probability $\sigma \in [0, 1]$. We can then guarantee expected cost

$$\max_{\tau \in [0,1]} V(\sigma, \tau) \;=\; \max\bigl\{\sigma A + (1-\sigma)C, \sigma B + (1-\sigma)D\bigr\}.$$

The right hand is a maximum of two linear functions. We now use (2.1). Since $(A - C)(B - D) < 0$ one function is increasing in σ while the other is decreasing, so their maximum is minimised at their intersection. Moreover, this intersection occurs for $\sigma \in (0, 1)$ because $(A - B)(C - D) < 0$. Hence the unique $\hat{\sigma}$ is found by solving for σ in

$$\sigma A + (1-\sigma)C \;=\; \sigma B + (1-\sigma)D.$$

The computation of $\hat{\tau}$ is analogous, and simple algebra yields $\mathcal{V}$. □

Equaliser Strategies The proof of the lemma in fact shows something more, namely that both $\hat{\sigma}$ and $\hat{\tau}$ are *equaliser strategies*, that is, $V(\hat{\sigma}, \tau) = V(\sigma, \hat{\tau}) = \mathcal{V}$ for all σ and τ. The equaliser strategy $\hat{\sigma}$ removes all power from Adversary by rendering all her strategies exactly equally costly for us.

Expectations We stress that all expectations in this chapter refer to strategic randomness deliberately introduced by the players to realise their guarantees. In particular, we never assume that outcomes are drawn from a true distribution.

With now apply this machinery to forecasting a binary outcome.

2.3 Predicting a Single Binary Outcome

We first model the single-trial problem as a game. Both we and Adversary choose a value in $\{\mathsf{y}, \mathsf{n}\}$. We call our value the *prediction* and Adversary's value the *actual outcome*. We make a mistake if our prediction does not equal the actual outcome. Recall that there are two experts, one that predicts y and one that predicts n. Since, in the single-trial case, the best expert makes no mistake, our regret equals our loss. Taking our regret as our cost results in the *one-shot regret game*:

$\mathcal{G} :=$

We \ Adversary	y	n
y	0	1
n	1	0

Both our pure strategies y and n guarantee that we make at most one mistake. This guarantee is trivial, since it is the maximal number of mistakes. We now show that randomisation improves our guarantee.

2.3.1. Theorem. *The one-shot regret game $\mathcal{G}$ has minimax regret $\mathcal{V}$ and unique saddle point $\langle \hat{\sigma}, \hat{\tau} \rangle$, where*

$$\mathcal{V} = 1/2, \qquad \hat{\sigma} = 1/2 \qquad \text{and} \qquad \hat{\tau} = 1/2.$$

Proof. By (2.1) $\mathcal{G}$ has no saddle point in pure strategies, and hence the unique saddle point and game value are given by Lemma 2.2.1. □

The strategy $\hat{\sigma}$, which predicts uniformly at random, is an equaliser strategy that guarantees that we make a mistake with probability exactly a half. That is, all actual outcomes are equally adversarial.

In a sense, predicting a single binary outcome with two experts is not very interesting because we have *too many* (2) experts relative to the number of outcomes (2): so many that the best expert always has loss

zero. The situation is very different if we predict a sequence of multiple binary outcomes in a row. We first introduce the necessary repeated-game theory and subsequently analyse the scenario with repetition.

2.4 Repeated Games

A repeated game is a matrix game that results in games [16]. Such games are defined recursively. In the elementary (zero-round) game $V \in \mathbb{R}$ we simply suffer cost V. Then if $\mathcal{H}, \mathcal{I}, \mathcal{J}$ and $\mathcal{K}$ are games, so is the composite game

	y	n
y	$\mathcal{H}$	$\mathcal{I}$
n	$\mathcal{J}$	$\mathcal{K}$

in which the two players independently choose an action, and jointly determine the subsequent game to be played. An example of a two-trial game is

		y			n	
		y	n		y	n
y	y	A	B	y	E	F
	n	C	D	n	G	H
		y	n		y	n
n	y	I	J	y	M	N
	n	K	L	n	O	P

. (2.2)

Say that, in the first trial, we play y and Adversary plays n. Together, these actions determine that the players subsequently face the constituent game

	y	n
y	E	F
n	G	H

. (2.3)

If, in the second trial, we play n and Adversary plays n, then the players subsequently face the elementary game

$$H, \tag{2.4}$$

in which we immediately incur the specified cost H.

A *history* of length t is an element of $\{\mathsf{y},\mathsf{n}\}^t \times \{\mathsf{y},\mathsf{n}\}^t$. A history $\langle p, x\rangle$ of length t records our prediction p_i and the actual outcome x_i for each played trial $1 \le i \le t$. The subgame of a game $\mathcal{G}$ identified by the history $\langle p, x\rangle$ is denoted by $\mathcal{G}_{p,x}$. For example, let $\mathcal{G}$ denote the game displayed as (2.2). The history $\langle \mathsf{y},\mathsf{n}\rangle$ identifies the subgame $\mathcal{G}_{\mathsf{y},\mathsf{n}}$ displayed as (2.3), while the history $\langle \mathsf{yn},\mathsf{nn}\rangle$ identifies the o-round subgame $\mathcal{G}_{\mathsf{yn},\mathsf{nn}}$ displayed as (2.4). The *longer* the history, the *shorter* the subgame it identifies. The empty history $\langle \epsilon, \epsilon\rangle$ identifies the entire game itself: $\mathcal{G} = \mathcal{G}_{\epsilon,\epsilon}$ for any game $\mathcal{G}$.

As before, we are looking for the optimal strategies and for the game value, i.e. the best possible expected cost guarantee. Repeated games can be solved recursively by the procedure called *backwards induction*, also known as *dynamic programming* or *Zermelo's algorithm* [16].

2.4.1 Backwards Induction

Solving elementary games is trivial. To solve a composite game

$$\mathcal{G} \;=\; \begin{array}{c|cc} & \mathsf{y} & \mathsf{n} \\ \hline \mathsf{y} & \mathcal{G}_{\mathsf{y},\mathsf{y}} & \mathcal{G}_{\mathsf{y},\mathsf{n}} \\ \mathsf{n} & \mathcal{G}_{\mathsf{n},\mathsf{y}} & \mathcal{G}_{\mathsf{n},\mathsf{n}} \end{array},$$

first recursively solve its four constituent games to obtain their best achievable expected cost guarantees $\mathcal{V}_{\mathsf{y},\mathsf{y}}$, $\mathcal{V}_{\mathsf{y},\mathsf{n}}$, $\mathcal{V}_{\mathsf{n},\mathsf{y}}$, and $\mathcal{V}_{\mathsf{n},\mathsf{n}}$. Second, replace each constituent game by its value to obtain the single-trial game

$$\mathcal{G}' \;=\; \begin{array}{c|cc} & \mathsf{y} & \mathsf{n} \\ \hline \mathsf{y} & \mathcal{V}_{\mathsf{y},\mathsf{y}} & \mathcal{V}_{\mathsf{y},\mathsf{n}} \\ \mathsf{n} & \mathcal{V}_{\mathsf{n},\mathsf{y}} & \mathcal{V}_{\mathsf{n},\mathsf{n}} \end{array}.$$

The game $\mathcal{G}'$ represents the game $\mathcal{G}$, assuming that we play all its subgames optimally. By solving the single-trial game $\mathcal{G}'$ using Section 2.2, we therefore obtain the minimax strategy $\hat{\sigma}$, maximin strategy $\hat{\sigma}$ and value $\mathcal{V}$ of the composite game $\mathcal{G}$.

Backwards induction recursively solves all subgames of $\mathcal{G}$. For each subgame $\mathcal{G}_{p,x}$, we denote its game value by $\mathcal{V}_{p,x}$ and its saddle point by $\langle \hat{\sigma}_{p,x}, \hat{\tau}_{p,x}\rangle$.

2.5 Predicting a Sequence of Binary Outcomes

We now formalise the recurring prediction problem as a repeated game. We consider the case were we sequentially predict T binary outcomes, for some known fixed time horizon T. Recall that we have two experts, one that always predicts y and one that always predicts n, and that our goal is to minimise our regret w.r.t. the best expert.

Fix a sequence of predictions p and of actual outcomes x, both of length T. Our cumulative loss L and the cumulative loss of the best expert L^* are measured in mistakes, i.e.

$$L(p,x) \coloneqq \sum_{t=1}^{T} \mathbf{1}_{p_t \neq x_t} \quad \text{and} \quad L^*(x) \coloneqq \min\left\{ \sum_{t=1}^{T} \mathbf{1}_{\mathsf{y} \neq x_t}, \sum_{t=1}^{T} \mathbf{1}_{\mathsf{n} \neq x_t} \right\},$$

and our regret is thus

$$L(p,x) - L^*(x). \tag{2.5}$$

The regret is always evaluated at sequences of length exactly T, but it is convenient to allow the cumulative losses L and L^* to be evaluated at all p and x of equal length $t \leq T$, with in particular $L(\epsilon,\epsilon) = L^*(\epsilon) = 0$.

2.5.1 The Regret Game with Time Horizon T

We now formalise the sequential prediction problem with two experts as a repeated game, that we call the *regret game with time horizon T*. We first define its subgames $\mathcal{G}_{p,x}$ recursively. To each history $\langle p,x \rangle$ of length $t = T$ we assign the elementary game in which we incur the regret

$$\mathcal{G}_{p,x} \coloneqq L(p,x) - L^*(x), \tag{2.6a}$$

while for each history $\langle p,x \rangle$ of length $t < T$ we simply play one trial

$$\mathcal{G}_{p,x} \coloneqq \begin{array}{c|cc} & \mathsf{y} & \mathsf{n} \\ \hline \mathsf{y} & \mathcal{G}_{p\mathsf{y},x\mathsf{y}} & \mathcal{G}_{p\mathsf{y},x\mathsf{n}} \\ \mathsf{n} & \mathcal{G}_{p\mathsf{n},x\mathsf{y}} & \mathcal{G}_{p\mathsf{n},x\mathsf{n}} \end{array}. \tag{2.6b}$$

The *regret game with time horizon T* starts from the empty history $\langle \epsilon,\epsilon \rangle$

$$\mathcal{G} \coloneqq \mathcal{G}_{\epsilon,\epsilon}. \tag{2.6c}$$

Predicting like the best expert amounts to solving the game $\mathcal{G}$. Our minimax strategy approximately predicts like the best expert, and our minimax regret quantifies the discrepancy. We solve the repeated game (2.6) by backwards induction. To simplify this task, we first use two properties of the specific cost function (2.6a), the regret, to simplify the representation of $\mathcal{G}$ and its subgames.

2.5.2 The Regret Game (Simplified)

The regret (2.5) is a special cost function. First, our cumulative loss L is a sum that decomposes over trials. Second, the cumulative loss L^* of the best expert does not depend on our predictions p. These two properties allow us to represent the game (2.6) more concisely by a game that is only parametrised by a sequence of actual outcomes x. The game $\mathcal{G}_x$ is defined as follows. To x of length $t = T$ we assign the elementary game with cost

$$\mathcal{G}_x \;:=\; -L^*(x), \tag{2.7a}$$

while for x of length $t < T$ we set[1]

$$\mathcal{G}_x \;:=\; \begin{array}{c|c|c} & \mathsf{y} & \mathsf{n} \\ \hline \mathsf{y} & \mathcal{G}_{x\mathsf{y}} & \mathcal{G}_{x\mathsf{n}} + 1 \\ \hline \mathsf{n} & \mathcal{G}_{x\mathsf{y}} + 1 & \mathcal{G}_{x\mathsf{n}} \\ \hline \end{array}. \tag{2.7b}$$

The *regret game with time horizon* T starts without observed outcomes

$$\mathcal{G} \;:=\; \mathcal{G}_\epsilon. \tag{2.7c}$$

In words, (2.7a) accounts for the cumulative loss of the best expert at the end of the game, while (2.7b) accounts for our mistakes (the $+1$) directly in the trial where they occur.

Both (2.6) and (2.7) define the regret game with time horizon T. A trivial induction on histories shows that they agree in the following sense.

[1] Incrementing a game $\mathcal{G}$ by $v \in \mathbb{R}$, denoted $\mathcal{G} + v$, means increasing all the costs of its elementary subgames by v. This does not affect the optimal strategy for either player, it simply increases the value of $\mathcal{G}$ by v.

2.5.1. Proposition. *For each history $\langle p, x\rangle$ of length $t \leq T$*

$$\mathcal{V}_{p,x} \;=\; \mathcal{V}_x + L(p,x), \qquad \hat{\sigma}_{p,x} \;=\; \hat{\sigma}_x \qquad \text{and} \qquad \hat{\tau}_{p,x} \;=\; \hat{\tau}_x,$$

where double subscripts refer to the regret game (2.6)*, while single subscripts refer to the simplified regret game* (2.7)*.*

In particular, the entire games $\mathcal{G}_{\epsilon,\epsilon}$ and $\mathcal{G}_\epsilon$ have the same value $\mathcal{V}_{\epsilon,\epsilon} = \mathcal{V}_\epsilon$, since $L(\epsilon,\epsilon) = 0$.

We are now ready to solve the simplified regret game with horizon T. We first solve its subgames locally in Section 2.5.3, then compute our minimax regret in Section 2.5.4, and finally implement our minimax strategy in Section 2.5.5. The next result is rather counter-intuitive.

2.5.3 Expertly Unpredictable Adversarial Outcomes

It follows from the analysis of the one-shot regret game in Section 2.3 that Adversary maximises our expected *cumulative loss* by drawing T outcomes uniformly at random. In the current game however, Adversary strives to maximise our expected *regret*. To this end, Adversary has to be simultaneously unpredictable to make our cumulative loss large and predictable to make the cumulative loss of the best expert small. These goals are obviously in direct conflict. Surprisingly, this discord is resolved optimally by dropping the latter goal. That is, Adversary also maximises our regret by drawing outcomes uniformly at random. This result has been noted earlier, for example in [25, Section 8.3], and holds quite generally. It also drives the minimax analyses [4] and [6].

2.5.2. Theorem. *The game $\mathcal{G}_x$ of* (2.7b) *has minimax regret $\mathcal{V}_x$ and a unique equaliser saddle point $\langle \hat{\sigma}_x, \hat{\tau}_x\rangle$ where*

$$\hat{\sigma}_x \;=\; \frac{\mathcal{V}_{xy} - \mathcal{V}_{xn} + 1}{2}, \quad \hat{\tau}_x \;=\; \frac{1}{2} \quad \text{and} \quad \mathcal{V}_x \;=\; \frac{\mathcal{V}_{xy} + \mathcal{V}_{xn} + 1}{2}.$$

Since $\hat{\tau}_x = 1/2$ independent of the past outcomes x, Adversary's maximin strategy draws outcomes uniformly at random each trial.

Proof. By backwards induction (Section 2.4.1), the solution of $\mathcal{G}_x$ equals the solution of the one-shot game of the values of its constituent games:

$$\mathcal{G}'_x \;=\; \begin{array}{c|cc} & \mathsf{y} & \mathsf{n} \\ \hline \mathsf{y} & \mathcal{V}_{x\mathsf{y}} & \mathcal{V}_{x\mathsf{n}} + 1 \\ \mathsf{n} & \mathcal{V}_{x\mathsf{y}} + 1 & \mathcal{V}_{x\mathsf{n}} \end{array}.$$

An easy induction on the length of x shows that $|\mathcal{V}_{x\mathsf{y}} - \mathcal{V}_{x\mathsf{n}}| \leq 1$. If $|\mathcal{V}_{x\mathsf{y}} - \mathcal{V}_{x\mathsf{n}}| < 1$, then $\mathcal{V}_x$ has no saddle point in pure strategies since

$$\big(\mathcal{V}_{x\mathsf{y}} - (\mathcal{V}_{x\mathsf{n}} + 1)\big)\big((\mathcal{V}_{x\mathsf{y}} + 1) - \mathcal{V}_{x\mathsf{n}}\big) \;=\; \big(\mathcal{V}_{x\mathsf{y}} - \mathcal{V}_{x\mathsf{n}}\big)^2 - 1 \;<\; 0$$

and

$$\big(\mathcal{V}_{x\mathsf{y}} - (\mathcal{V}_{x\mathsf{y}} + 1)\big)\big((\mathcal{V}_{x\mathsf{n}} + 1) - \mathcal{V}_{x\mathsf{n}}\big) \;=\; -1 \;<\; 0$$

verify condition (2.1), so that Lemma 2.2.1 yields the unique equaliser saddle point above. On the other hand if $|\mathcal{V}_{x\mathsf{y}} - \mathcal{V}_{x\mathsf{n}}| = 1$ then either

$$\mathcal{G}'_x \;=\; \begin{array}{c|cc} & \mathsf{y} & \mathsf{n} \\ \mathsf{y} & 0 & 0 \\ \mathsf{n} & +1 & -1 \end{array} + \mathcal{V}_{x\mathsf{y}} \qquad \text{or} \qquad \mathcal{G}'_x \;=\; \begin{array}{c|cc} & \mathsf{y} & \mathsf{n} \\ \mathsf{y} & -1 & +1 \\ \mathsf{n} & 0 & 0 \end{array} + \mathcal{V}_{x\mathsf{n}}.$$

In both cases the minimax strategy $\hat{\sigma}_x$ above is a unique, pure equaliser. In the left case, the entire interval $[1/2, 1]$ is maximin, while in the right case the entire interval $[0, 1/2]$ is maximin. In both cases, the unique equaliser maximin saddle point is $\hat{\tau}_x = 1/2$. □

Theorem 2.5.2 solves the game $\mathcal{G}_x$ in terms of the solutions to its constituent games. We now obtain a direct expression for our minimax regret of the full game $\mathcal{V}_\epsilon$.

2.5.4 Our Minimax Regret

Theorem 2.5.2 shows that, in the worst case, Adversary generates the T actual outcomes x uniformly at random. Then our expected cumulative loss is $T/2$, whatever our strategy. The expected loss of each individual expert is also $T/2$, and neither individual expert has any predictive value. Perhaps paradoxically, the expected cumulative loss of the *best* expert, i.e.

$$\mathbb{E}\big[L^*(x)\big] \;=\; \sum_{i=0}^{T} 2^{-T} \binom{T}{i} \min\{T - i, i\},$$

is strictly lower than $T/2$, as can be seen by applying Jensen's inequality to the concave min function. Our minimax regret is exactly the expected amount by which, purely by chance, the best expert beats just guessing.

2.5.3. THEOREM. *Our minimax regret $\mathcal{V}$ in the regret game with time horizon T is sandwiched as follows*

$$\sqrt{\frac{T-1}{2\pi}} \;\leq\; \mathcal{V} \;\leq\; \sqrt{\frac{T+1}{2\pi}} \tag{2.8}$$

A somewhat cruder asymptotic analysis can be found surrounding [25, Lemma 8.2].

Proof. Recall that our expected cumulative loss is $T/2$. We expand

$$\begin{aligned}
\mathcal{V} \;&=\; T/2 - \mathbb{E}\big[L^*(x)\big] \;=\; \sum_{i=0}^{T} 2^{-T}\binom{T}{i}\big(i - \min\{T-i, i\}\big) \;= \\
&\sum_{i=\lceil T/2\rceil}^{T} 2^{-T}\binom{T}{i}\big(i-(T-i)\big) \;=\; \sum_{i=\lceil T/2\rceil}^{T} 2^{-T} T\left(\binom{T-1}{i-1} - \binom{T-1}{i}\right) \\
&\overset{\text{telescope}}{=}\; 2^{-T} T\binom{T-1}{\lceil T/2\rceil - 1} \;=\; 2^{-T}\begin{cases} \dfrac{T!}{\frac{T-1}{2}!\frac{T-1}{2}!} & \text{if } T \text{ odd,} \\[2ex] \dfrac{T!}{\frac{T}{2}!\frac{T-2}{2}!} & \text{if } T \text{ even.} \end{cases}
\end{aligned} \tag{2.9}$$

This elegant telescoping argument goes back to [58]. We now show that

$$\sqrt{\frac{T-1}{2\pi}} \;<\; \frac{\Gamma\big(\frac{T+1}{2}\big)}{\Gamma\big(\frac{T}{2}\big)\Gamma\big(\frac{1}{2}\big)} \;\leq\; \mathcal{V} \;\leq\; \frac{\Gamma\big(\frac{T}{2}+1\big)}{\Gamma\big(\frac{T+1}{2}\big)\Gamma\big(\frac{1}{2}\big)} \;<\; \sqrt{\frac{T+1}{2\pi}}.$$

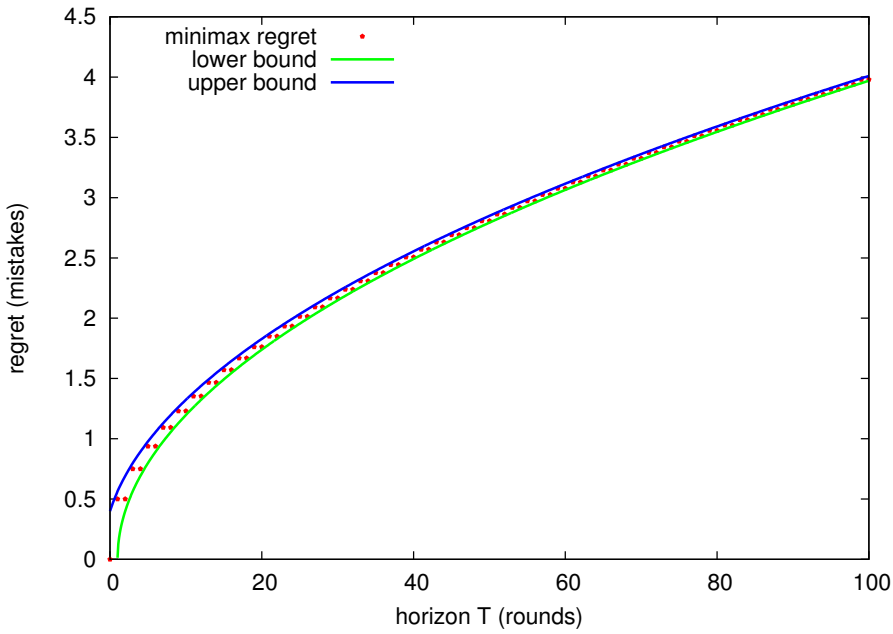
The innermost two inequalities follow from log-convexity of Γ which implies that the expression for odd T in (2.9) is a valid upper bound for even T, whereas the expression for even T in (2.9) is a valid lower bound for odd T. The outermost inequalities use $\Gamma(1/2) = \sqrt{\pi}$, and

$$\frac{\Gamma(x+1/2)}{\Gamma(x)}\frac{\Gamma(x+1)}{\Gamma(x+1/2)} \;=\; x,$$

where the smaller left factor must be $< \sqrt{x}$, and the right $> \sqrt{x}$. This implies that $\sqrt{x} < \Gamma(x+1)/\Gamma(x+1/2) < \sqrt{x+1/2}$ for all $x \geq 0$. □

Figure 2.1 displays our minimax regret in the regret game with time horizon T as a function of T, together with the two bounds of (2.8). We see that the bounds are good approximations even for small T.

Figure 2.1 Minimax regret of the regret game with time horizon T, shown as a function of time horizon T



Our minimax regret $\mathcal{V}$ grows sublinearly in the number of trials T, which means that our minimax strategy $\hat{\sigma}$ indeed learns to predict approximately like the best expert. In particular, when the best expert has small cumulative loss, so do we. Our minimax strategy thus attains the goal set out in the introduction. We now describe how to implement our minimax strategy $\hat{\sigma}$ efficiently.

2.5.5 Executing Our Minimax Strategy

So far, we saw that a strategy achieving the minimax regret $\mathcal{V}$ exists, and we found the recursive expression $\hat{\sigma}_x = (\mathcal{V}_{xy} - \mathcal{V}_{xn} + 1)/2$ in Theorem 2.5.2. In practise, to follow the minimax strategy $\hat{\sigma}_x$, we need an *efficient* method to sample according to it. Unfortunately, the recursive expression does not simplify to a direct expression for $\hat{\sigma}_x$. There are essentially three ways to still implement $\hat{\sigma}_x$.

2.5.5.1 Computing $\hat{\sigma}_x$ Exactly Using the Recursive Expression

Our minimax strategy $\hat{\sigma}_x$ is a function of a sequence of outcomes x, but it is clear that $\hat{\sigma}_x$ only depends on x via the current cumulative loss of both experts. Thus to compute $\hat{\sigma}_x$ for all x, it suffices to compute $\hat{\sigma}_x$ for the $O(T^2)$ possible pairs of cumulative losses of the two experts. This can be done in $O(T^2)$ time using $O(T^2)$ memory. We used this method to compute Figure 2.2, which displays our minimax strategy for time horizon $T = 101$. The colour values indicate the probability of predicting y. The optimal probability of predicting y has a sharp transition from 0 to 1 around the point where both experts have equal cumulative loss.

The advantage of this method it that it computes $\hat{\sigma}_x$ exactly for each x, as required e.g. for graphing. The disadvantage is that the exponent in the running time grows with the number of experts and outcomes, rendering it impractical for problems with many experts. We may give up computing $\hat{\sigma}_x$ exactly, and instead approximate it.

2.5.5.2 Computing $\hat{\sigma}_x$ Approximately

In Theorem 2.5.3 we sandwiched the minimax regret $\mathcal{V}$ tightly around $\sqrt{T/(2\pi)}$. A similar approach can be undertaken to bound the value $\mathcal{V}_x$ of each subgame $\mathcal{G}_x$, and then approximate $\hat{\sigma}_x$ in terms of these bounds.

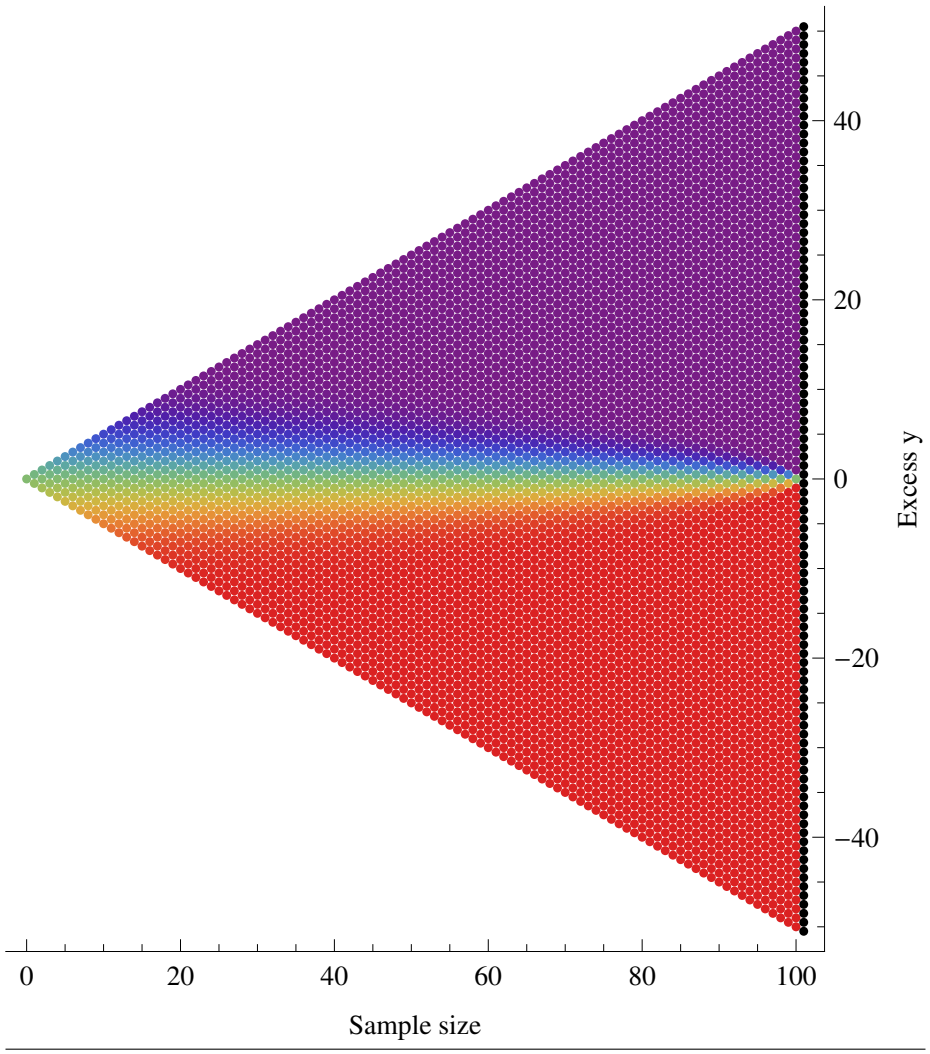
We have not explored this approach in detail, but it is clear that such methods can be much faster than $O(T)$ per trial, and scale better with the number of experts and outcomes. The difficulty lies of course in bounding the additional regret incurred by approximating. There is however a different method, that implements the minimax strategy exactly without computing $\hat{\sigma}_x$ exactly.

2.5.5.3 Not Computing $\hat{\sigma}_x$ At All

Interestingly, following the minimax strategy does not *require* computation of $\hat{\sigma}_x$, it requires predicting y with probability $\hat{\sigma}_x$. We now show that this can be done without computing $\hat{\sigma}_x$. In essence, we can think of our method as performing a randomised exact computation of $\hat{\sigma}_x$.

As proved in Theorem 2.5.2, in game $\mathcal{G}_x$ our minimax strategy predicts y with probability $\hat{\sigma}_x = (\mathcal{V}_{x\mathsf{y}} - \mathcal{V}_{x\mathsf{n}} + 1)/2$. We now devise a

Figure 2.2 Minimax strategy in the regret game with time horizon $T = 101$. The colour value at position (t, e) indicates the probability $\hat{\sigma}$ that our minimax strategy assigns to observing y next after t trials with $t/2 + e$ occurrences of outcome y and hence $t/2 - e$ occurrences of outcome n. The probabilities range from 0 (red) via 1/2 (green) to 1 (violet). The game has ended in black dots.



Algorithm 2.1 Minimax algorithm for time horizon T

Input: Game $\mathcal{G}_x$ with observed past outcomes x of length $t < T$.
Sample a continuation z uniformly at random from $\{\mathsf{y},\mathsf{n}\}^{T-t-1}$.
Compute $\mathcal{V}_{x\mathsf{y}z} = -L^*(x\mathsf{y}z)$ and $\mathcal{V}_{x\mathsf{n}z} = -L^*(x\mathsf{n}z)$.
Predict y with probability $(\mathcal{V}_{x\mathsf{y}z} - \mathcal{V}_{x\mathsf{n}z} + 1)/2$.

method to generate such predictions without computing $\mathcal{V}_{x\mathsf{y}}$ and $\mathcal{V}_{x\mathsf{n}}$ exactly. Our algorithm is displayed as Algorithm 2.1. In words, the algorithm takes $x\mathsf{y}$ and $x\mathsf{n}$, the two possible one-step extensions of the observed actual outcomes x, completes them both to length T by appending the same random future outcomes z, and plays the outcome whose extension incurs the smaller regret. If both completions incur the same regret, it predicts by flipping a fair coin.

This algorithm is new to the best of our knowledge. It interpolates between the algorithms in [4] and [6], in the sense that it uses random sampling akin to [6], but instead of using a random future to estimate the best expert, it is used to gauge the quality of both outcomes as in the recurrence relations in [4].

We now analyse the algorithm. First we prove correctness, and then consider efficiency.

2.5.4. Theorem. *Algorithm 2.1 implements our minimax strategy $\hat{\sigma}$.*

Proof. By Theorem 2.5.2, for any history x of length t and z sampled uniformly at random from $\{\mathsf{y},\mathsf{n}\}^{T-t}$, we have $\mathbb{E}[\mathcal{V}_{xz}] = (T-t)/2 + \mathcal{V}_x$. Since the sequences $x\mathsf{y}z$ and $x\mathsf{n}z$ differ only in one position, the cumulative loss of the best expert differs by at most one, so that $\left|\mathbb{E}[\mathcal{V}_{x\mathsf{y}z}] - \mathbb{E}[\mathcal{V}_{x\mathsf{n}z}]\right| \leq 1$. The probability that the algorithm predicts y is therefore given by

$$\mathbb{E}\left[\frac{\mathcal{V}_{x\mathsf{y}z} - \mathcal{V}_{x\mathsf{n}z} + 1}{2}\right] = \frac{\mathbb{E}[\mathcal{V}_{x\mathsf{y}z} - \mathcal{V}_{x\mathsf{n}z}] + 1}{2} = \frac{\mathcal{V}_{x\mathsf{y}} - \mathcal{V}_{x\mathsf{n}} + 1}{2} = \hat{\sigma}_x$$

as desired. □

Performing this procedure in trial t takes $O(T-t)$ steps and uses $O(T-t)$ memory. Predicting T outcomes thus takes $O(T^2)$ time in total. Since the memory can be reused, $O(T)$ memory suffices.

The advantage of this method is that it is extremely simple, and that it scales to more difficult prediction problems extremely well, as will be explored in Section 2.8. For our two constant experts, the effect of appending a random future is that the current cumulative losses of both experts are incremented by a binomial random variable. Drawing a binomial random variable can be done more efficiently, see e.g. [87].

Another interesting possibility is to re-use a single randomly drawn full future of length T, reducing the running time to $O(T)$ in total. This makes our predictions in subsequent trials statistically dependent, and this can be used against us by Adversary. However, for so-called *oblivious adversaries* [25], i.e. adversaries that do not look at our predictions at all, this fast strategy still attains the minimax regret.

2.6 Variations on a Theme

We modelled predicting a binary outcome as the regret game with time horizon T, quantified the best achievable regret, and implemented our minimax strategy efficiently. We now extend our results to a variety of similar prediction problems, simultaneously obtaining near-optimal solutions to the corresponding regret games, and resulting in new insights into our minimax strategy.

2.6.1 Stopping Early

We now consider what happens if we give Adversary the power to stop the game early, before the time horizon T is reached. That is, at the beginning of each trial, we allow Adversary to declare the game finished, and if she does we suffer the regret at that time. That is, if Adversary stops in history $\langle p, x\rangle$, we suffer $L(p, x) - L^*(x)$. If Adversary chooses to continue, then the game proceeds as before. We now show that this extra power cannot be used to beat Theorem 2.5.3.

2.6.1. Theorem. *Our minimax regret without and with adversarial stopping are identical.*

Proof. We show that stopping is a dominated strategy. Say that we are in some history $\langle p, x\rangle$, and assume w.l.o.g. that constant-y is the best expert. By stopping, Adversary inflicts regret $L(p, x) - L^*(x)$. However, she is better off by playing outcome n until time T. This will not

change the cumulative loss of the best expert, nor will it decrease our cumulative loss. Hence our eventual regret is at least our current regret. Thus stopping is never useful. □

Note that we do not claim that our minimax regret (2.8) now can be evaluated with the actual stopping time substituted for the original time horizon T. We simply say that our minimax regret with the original time horizon T remains unchanged. We now obtain a bound that can be evaluated on the actual stopping time.

2.6.2 Unknown Time Horizon T

It is not always realistic to assume that the number of trials T is known beforehand. Still, it is possible to approximately achieve the minimax regret (2.8) without knowing T by a method called the *doubling trick* [25]. Start by segmenting the trials as follows.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	⋯

Play the first trial as if $T = 1$. Then reset the algorithm and play the next two trials as if $T = 2$. Again, reset the algorithm and play the subsequent four trials as if $T = 4$ etc. Each reset completely restarts the algorithm, ignoring the history generated until then. This procedure guarantees that in segment i (of length 2^{i-1}), our regret is at most $\sqrt{(2^{i-1}+1)/(2\pi)}$ w.r.t. the best expert on that segment. The cumulative loss of the segmentwise best expert is at most the cumulative loss of the best expert. To analyse the regret, we first prove the following intermediate result.

2.6.2. Lemma. *For any natural number b*

$$\sum_{i=1}^{b} \sqrt{2^{i-1}+1} \;\le\; (1+\sqrt{2})\sqrt{2^b}.$$

Proof. By concavity of the square root $\sqrt{x+1} \le \sqrt{x} + \frac{1}{2\sqrt{x}}$, so that

$$\begin{aligned}\sum_{i=1}^{b} \sqrt{2^{i-1}+1} \;&\le\; \sum_{i=1}^{b} \left(\sqrt{2^{i-1}} + \frac{1}{2\sqrt{2^{i-1}}}\right) \;=\; \frac{\sqrt{2^b}-1}{\sqrt{2}-1} + \frac{1-\sqrt{2^{-b}}}{2-\sqrt{2}} \\ &=\; (1+\sqrt{2})\left(\sqrt{2^b} - 1 + 1/\sqrt{2} - \sqrt{2^{-b}}/\sqrt{2}\right).\end{aligned}$$

Then observe that $-1 + 1/\sqrt{2} - \sqrt{2^{-b}}/\sqrt{2} \le 0$ for all b. □

This lemma allows us to analyse the guarantee provided by the doubling trick. If the actual length is T, we use $\lceil \log_2(T+1) \rceil$ segments. The last segment may be stopped early, but by the previous section the full regret bound still holds there. In total, we thus guarantee regret

$$\sum_{i=1}^{\lceil \log_2(T+1) \rceil} \sqrt{\frac{2^{i-1}+1}{2\pi}} \;\leq\; (1+\sqrt{2})\sqrt{2^{\lceil \log_2(T+1) \rceil}} \;\leq\; (2+\sqrt{2})\sqrt{\frac{T+1}{2\pi}}$$

uniformly over time. The first inequality is Lemma 2.6.2, and the second uses $\lceil x \rceil \leq x+1$.

This bound shows that not knowing the time horizon T multiplies the regret (2.8) by at most a factor $2+\sqrt{2} \approx 3.414$. This is close to the slightly better factor $2\sqrt{\pi \ln(2)} \approx 2.951$ that is achieved by the exponentially weighted average forecaster with suitably decreasing learning rate [25, Theorem 2.3].

2.6.3 Two Adversarial Experts

In this section we replace our two constant experts with two adversarial or black-box experts (see the taxonomy in Section 1.6), whose predictions are determined by Adversary. It is clear that this increases Adversary's power, and hence increases our minimax regret. We now show that our regret remains as it is, that is, constant experts are the worst case.

First observe that trials in which the experts agree are wasted by Adversary. Namely, in such trials we can ensure that our regret does not grow, by issuing the same prediction as both experts. Since such trials do not influence our regret, Adversary may as well delay them to the end of the game. But then they amount to early stopping, which we proved suboptimal in Section 2.6.1.

Second, observe that we can interchange the labels y and n (by flipping both predictions and outcomes) while preserving the instantaneous loss of each prediction/outcome pair. Hence we may assume that one expert always predicts y, while the other always predicts n.

Combining disagreement and label interchange, we see that our original two constant experts are actually adversarial, and that the minimax algorithm for constant experts can be transformed into the minimax algorithm for adversarial experts by reinterpreting $\hat{\sigma}_x$ as the probability of following the first expert.

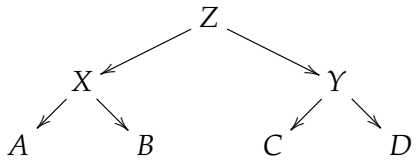
2.6.4 Many Experts

We have an algorithm for combining the binary predictions of two experts. Faced with many experts, an obvious way to combine their (still binary) predictions is to combine them in a binary tree.

2.6.3. THEOREM. *The algorithm that combines n experts recursively in a binary tree, playing the minimax strategy for adversarial experts at each internal node, guarantees expected regret at most*

$$\lceil \log(n) \rceil \sqrt{\frac{T+1}{2\pi}}.$$

Proof. We prove the simple case of four experts. Let's call them A, B, C and D. We recursively combine them as follows



Here the leaves A, B, C and D are basic experts, and the internal meta-experts X, Y and Z represent experts that predict by running the minimax algorithm on their respective children. To analyse the regret of Z w.r.t. the best basic expert, we abbreviate the cumulative loss of each expert $\xi \in \{A, B, C, D, X, Y, Z\}$ by L_ξ, and denote the regret incurred by the meta-experts X, Y and Z w.r.t. their children by R_X, R_Y and R_Z. The cumulative losses and regrets of the meta-experts are random variables, determined by the internal randomisation of the minimax algorithm. We have

$$\begin{aligned}
\mathbb{E}[L_Z] &= \mathbb{E}\big[\min\{L_X, L_Y\} + R_Z\big] \\
&= \mathbb{E}\Big[\min\big\{\min\{L_A, L_B\} + R_X, \min\{L_C, L_D\} + R_Y\big\}\Big] + \mathbb{E}[R_Z] \\
&\leq \min\big\{\min\{L_A, L_B\} + \mathbb{E}[R_X], \min\{L_C, L_D\} + \mathbb{E}[R_Y]\big\} + \mathbb{E}[R_Z] \\
&\leq \min\{L_A, L_B, L_C, L_D\} + 2\sqrt{\frac{T+1}{2\pi}}
\end{aligned}$$

by applying Jensen's inequality to the concave function min, and using the minimax regret bound Theorem 2.5.3. The case for n experts is analogous. □

Expected regret that depends logarithmically on the number of experts n is already quite good, but even better guarantees can be obtained. The exponentially weighted average forecaster [25, Theorem 2.2] achieves expected regret bounded by

$$\sqrt{\frac{T}{2}\ln(n)},$$

with square-root-of-ln dependence on the number of experts n.

2.7 Good Best Expert

Let's consider again the simplest case, prediction with two constant experts. Our minimax strategy for the regret game with time horizon T guarantees the minimax regret $\sqrt{(T+1)/(2\pi)}$ w.r.t. the best expert, irrespective of Adversary's strategy. In fact, our minimax strategy equalises the expected regret over all possible strategies for Adversary. That is, we suffer the same expected regret whatever the cumulative loss of the best expert. There are no sequences of outcomes for which we are *lucky* in the sense that our expected regret is actually smaller than the bound indicates.

The concept of *luckiness* [163, 71]is the idea that we desire to incur tiny regret whenever the best expert has small cumulative loss, and to achieve this, are willing to accept a slight increase in overhead when the best expert has moderate or large cumulative loss.

In this section we investigate a fundamentally different strategy, namely the strategy that enforces maximal luck. To this end, we first assume that we know that the best expert makes at most k mistakes. We then create a new prediction game where this assumption is built into the rules, i.e. constrains Adversary's moves. As before, we find the minimax strategy in this prediction game. Finally, we get rid of this assumption using the doubling trick (as in Section 2.6.2).

The constraint that the cumulative loss of the best expert is at most k serves as a "loss horizon", replacing the role of the time horizon T. The minimax strategy and regret that we obtain in this section are therefore parametrised by k instead of T.

2.7.1 The Regret Game With Loss Horizon k

The *regret game with loss horizon k* is defined as follows. In trial t, we pick a prediction p_t and Adversary plays an actual outcome x_t. As before, both moves are randomised independently. If in any history $\langle p, x \rangle$ the best expert has cumulative loss $L^*(x) > k$, the game ceases and we suffer regret $-\infty$. This is so bad for Adversary that she will never let this happen. Otherwise, the game continues for infinitely many trials. In the resulting infinite history $\langle p, x \rangle$ we suffer the regret $L(p, x) - L^*(x)$. We may allow Adversary to stop the game early, but this is a dominated strategy that does not increase her power, by the same reasoning as in Section 2.6.1. We first simplify the game, then solve it.

2.7.2 Simplified Regret Game

Infinite plays of the regret game with loss horizon k are pretty boring apart from a finite initial part of length at most $2k + 1$. This is because after at most $2k + 1$ outcomes, one expert must have cumulative loss $k + 1$, meaning that this expert cannot be the eventual best expert. We say that such an expert is *dead*.

We now show that our minimax strategy never follows a dead expert. This in turn means that once an expert dies, the regret never changes. We and the best expert either both or neither incur loss.

Say w.l.o.g. that the constant-n expert is dead. Then both our cumulative loss and the cumulative loss of the best expert decompose over trials. That is, each such trial we are facing the game that changes our regret as follows:

	y	n
y	0	0
n	$+1$	-1

This game has a saddle point in pure strategies, $\hat{\sigma} = \hat{\tau} = 1$, and game value $\mathcal{V} = 0$. Again $\hat{\sigma}$ is an equaliser strategy: by deterministically playing y, we guarantee that our eventual regret equals our current regret $L(p, x) - L^*(x)$, irrespective of Adversary's strategy.

The regret game with loss horizon k is quite different from the regret game with time horizon T. Surprisingly, we now solve the former game problem by reducing it to the latter.

2.7.3 Reduction to Time Horizon $T = 2k + 1$

We argued that once an expert is dead, our minimax strategy ensures that the eventual regret equals the current regret. So we might as well stop the game then or at any later time. We propose to stop the regret game with loss horizon k at time $T = 2k + 1$. We know that at this time exactly one expert is dead, so the game can safely be stopped. Note in particular that the rule that hands out $-\infty$ regret when both experts die cannot yet have been invoked. Vice versa, in the regret game with time horizon $T = 2k + 1$, the best expert will suffer loss at most k. But this means that these games are identical, and hence share the same value and optimal strategies. In particular

2.7.1. THEOREM. *The regret game with loss horizon k has minimax regret*

$$\sqrt{\frac{k}{\pi}} \;\leq\; \mathcal{V} \;\leq\; \sqrt{\frac{k+1}{\pi}}. \tag{2.10}$$

Adversary's maximin strategy samples outcomes uniformly at random until one expert dies. After that, she always chooses the outcome predicted by the live expert. When both experts are still alive, our minimax strategy is the one described in Section 2.5.5. After one expert has died, we follow the live expert.

Proof. By Theorem 2.5.3. □

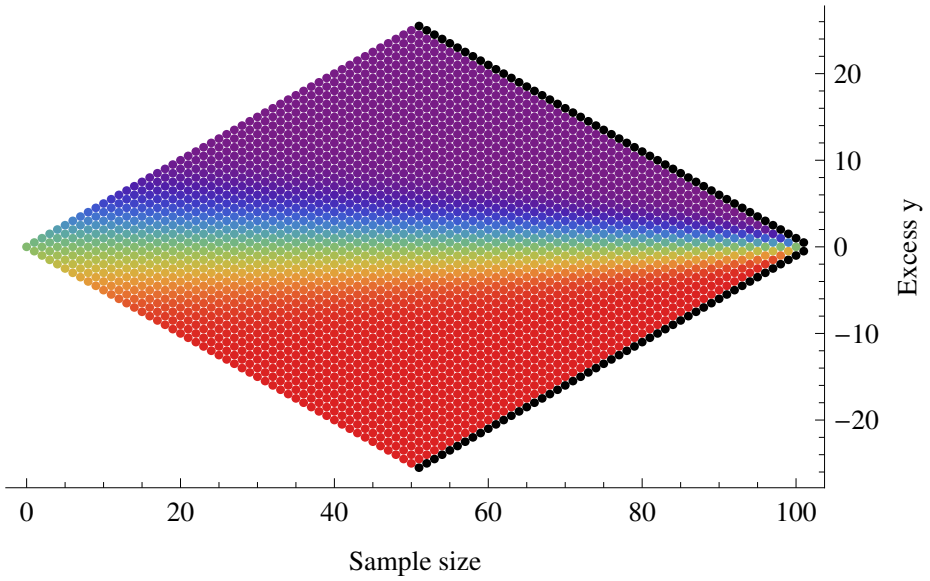
The equivalence of a loss horizon with a particular time horizon has not been noted in the literature, perhaps because it is most apparent for two experts. A generalisation to many experts can undoubtedly be obtained, but falls outside the scope of this minimalistic introduction.

Our minimax strategy is displayed for $k = 50$ in Figure 2.3. This figure is a truncated version of the optimal strategy with time horizon $T = 2k + 1 = 101$, which is shown in Figure 2.2.

2.7.4 Unknown Loss Horizon k

In Section 2.6.2 we eliminated the need to know the time horizon T by using the doubling trick. Here we face the problem that we may not know k. Again we apply the doubling trick. We run our minimax algorithm assuming that $k = 1$. When the worst expert dies, the game with this assumption is essentially over, and we restart completely, now

Figure 2.3 Minimax strategy in the regret game with loss horizon $k = 50$. This is a subplot of Figure 2.2. The colour value at position (t, e) indicates the probability $\hat{\sigma}$ that our minimax strategy assigns to observing y next after t trials with $t/2 + e$ occurrences of outcome y and hence $t/2 - e$ occurrences of outcome n. The game has (essentially) ended in black dots, since one expert is dead.



with $k = 2$. Again when the worst expert dies we restart with doubled loss horizon. The regret guarantee becomes

$$\sum_{i=1}^{\lceil \log_2(k+1) \rceil} \sqrt{\frac{2^{i-1}+1}{\pi}} \;\leq\; (1+\sqrt{2})\sqrt{2^{\lceil \log_2(k+1) \rceil}} \;\leq\; (2+\sqrt{2})\sqrt{\frac{k+1}{\pi}}.$$

The first inequality is Lemma 2.6.2, and the second uses $\lceil x \rceil \leq x + 1$.

This bound shows that not knowing the loss horizon k multiplies the regret (2.10) by at most a factor $2 + \sqrt{2} \approx 3.414$.

We now leave our two constant experts, to compete with more complicated white-box experts.

2.8 Competing with a 1-Lipschitz Best Expert

We now consider prediction with more complicated white-box experts (see Section 1.6). In fact, we abstract so dramatically that the experts almost disappear. Fix a time horizon T. We take as our basic ingredient a function $L^*\colon \{\mathsf{y},\mathsf{n}\}^T \to \mathbb{R}$ that measures the cumulative loss of the best expert.

The *L^*-regret game with time horizon T* is obtained by substituting the given function L^* into defining equation (2.7) on page 30. That is, the game structure remains the full binary tree of depth T, but the payoff is now the regret w.r.t. the best expert as specified by L^*.

In this section, we assume that L^* is 1-Lipschitz in the Hamming metric, which means that $L^*(x)$ changes by at most 1 when we flip one outcome of x. This assumption obviously holds for all L^* that we have considered so far, since the cumulative loss of *any* static (see Section 1.6) expert changes by at most one if a single outcome is flipped, and hence so does the cumulative loss of the best expert.

By choosing an intricate best expert cumulative loss function L^*, we obtain interesting prediction games. The many-expert bounds and algorithms that we saw in Section 2.6.4 are tight for adversarial experts, that is, assuming that Adversary can control the loss of each expert independently, or at least approximately so. But the experts underlying L^* can be quite far from independent, implying that the minimax algorithm is fundamentally different, and guarantees smaller regret than the adversarial bounds suggest.

We will consider the particular application of switching experts (or tracking) in Section 2.9. Interestingly and curiously, it is possible to obtain a single minimax strategy, parametrised by L^*, that works for all 1-Lipschitz L^*.

2.8.1 The Minimax Strategy

We now show that Algorithm 2.1 remains our minimax strategy in the L^*-regret game with time horizon T. As far as we know this unexpected result is new. We also show that, as before, Adversary's maximin strategy draws outcomes uniformly at random. The crucial observation is that the solution of the vanilla regret game with time horizon T, given as Theorem 2.5.2, applies to the more general L^*-regret game.

2.8.1. Theorem. *Fix a 1-Lipschitz function L^*, time horizon T, and let $\mathcal{G}$ be the L^*-regret game with time horizon T as defined in equation (2.7) on page 30. Then for any sequence of outcomes x of length $t < T$, the game $\mathcal{G}_x$ has minimax regret $\mathcal{V}_x$ and a unique equaliser saddle point $\langle \hat{\sigma}_x, \hat{\tau}_x \rangle$ where*

$$\hat{\sigma}_x \;=\; \frac{\mathcal{V}_{x\mathsf{y}} - \mathcal{V}_{x\mathsf{n}} + 1}{2} \qquad \hat{\tau}_x \;=\; \frac{1}{2} \qquad \mathcal{V}_x \;=\; \frac{\mathcal{V}_{x\mathsf{y}} + \mathcal{V}_{x\mathsf{n}} + 1}{2}$$

Again, Adversary's maximin strategy is to play uniformly at random. Note that these recurrence relations are exactly the same as obtained in Theorem 2.5.2. However, the different base case $\mathcal{V}_x = -L^*$ for final x of length T, affects all derived values and minimax strategies.

Proof. The critical inequality is to show $|\mathcal{V}_{x\mathsf{y}} - \mathcal{V}_{x\mathsf{n}}| \leq 1$, the remainder of the proof equals that of Theorem 2.5.2. We prove this inequality by induction on the length t of x. With z denoting a sequence of $T - t - 1$ outcomes drawn uniformly at random,

$$\mathcal{V}_{x\mathsf{y}} - \mathcal{V}_{x\mathsf{n}} \;=\; \mathbb{E}\big[\mathcal{V}_{x\mathsf{y}z} - \mathcal{V}_{x\mathsf{n}z}\big] \;=\; \mathbb{E}\big[-L^*(x\mathsf{y}z) + L^*(x\mathsf{n}z)\big].$$

By Jensen's inequality applied to the convex absolute-value function, and by using that L^* is 1-Lipschitz we obtain

$$|\mathcal{V}_{x\mathsf{y}} - \mathcal{V}_{x\mathsf{n}}| \;\leq\; \mathbb{E}\Big[\big|-L^*(x\mathsf{y}z) + L^*(x\mathsf{n}z)\big|\Big] \;\leq\; 1. \qquad \square$$

It follows that Algorithm 2.1 implements our minimax strategy. The efficiency of this algorithm now depends on the resources required to evaluate L^*. It also follows that the game value equals

$$\mathcal{V} \;=\; T/2 - \mathbb{E}\big[L^*(x)\big].$$

We now consider a simple example.

2.8.2 Example

To illustrate Theorem 2.8.1, we now work through a simple example. We consider predicting $T = 3$ outcomes, and compete with all experts that predict y exactly once. That is, our goal is to learn which outcome is the y. In formulas, our set of experts is

$$\Xi \;:=\; \{\mathsf{ynn}, \mathsf{nyn}, \mathsf{nny}\},$$

so that the cumulative loss of the best expert is

$$L^*(x) \coloneqq \min_{\xi \in \Xi} L(\xi, x).$$

The function L^* is clearly 1-Lipschitz, since the cumulative loss of *any* fixed sequence of predictions changes by at most one if a single outcome is flipped, and hence so does the cumulative loss of the best expert. We call this particular regret game the Ξ-*regret game*.

Figure 2.4 displays the solution, obtained by backwards induction (Section 2.4.1), to the Ξ-regret game. The figure indicates the value $\mathcal{V}_x$ of each subgame $\mathcal{G}_x$, and either the best expert(s) when the game is over or the minimax strategy $\hat{\sigma}_x$ when trials remain. We see that the minimax regret of the full game is $3/4$, and that this equals our expected cumulative loss, which is $T/2 = 3/2$, minus the expected cumulative loss of the best expert, which is $(2+1+1+0+1+0+0+1)/8 = 3/4$.

We see that, starting at $\hat{\sigma} = 1/4$, our minimax strategy doubles the probability of predicting y as long as ns are observed. This is reminiscent of the investment strategy underlying the St. Petersburg paradox. After the first y, an eventual best expert is revealed, and we deterministically predict n.

This was a simple example. An important choice of L^* arises from considering as experts all prediction sequences with few switches. We now consider this application.

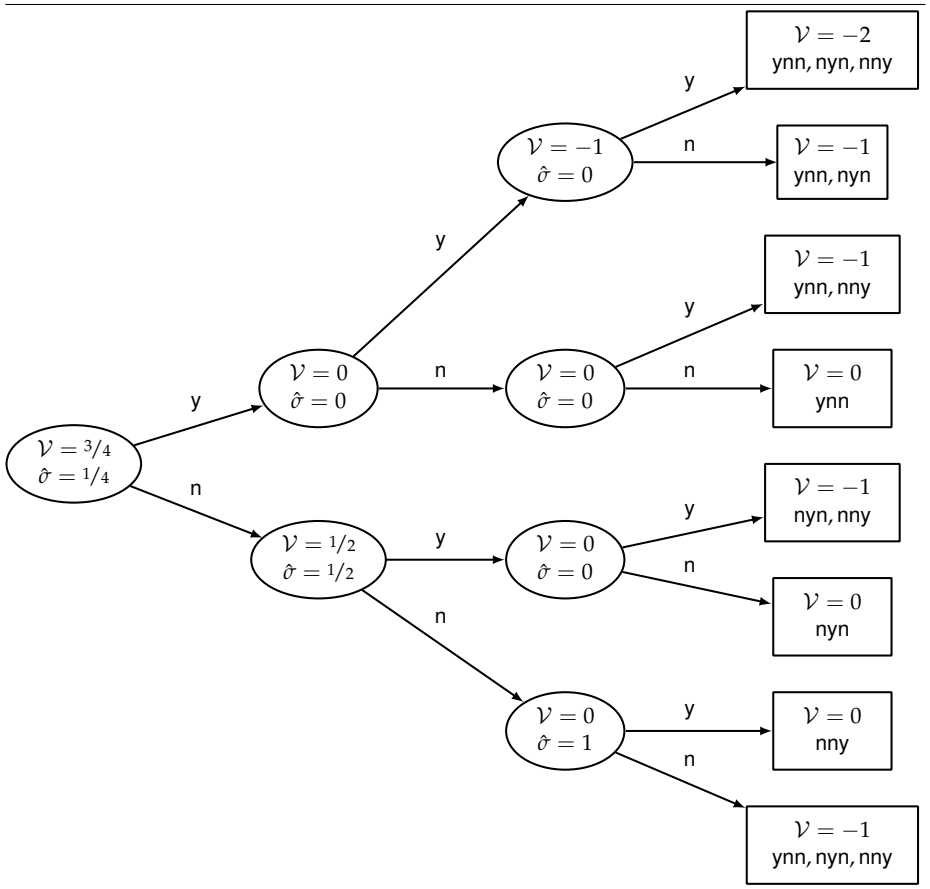
2.9 Switching

In this section, we apply the theory of competing with a 1-Lipschitz best expert to prediction with experts that divide the outcomes in blocks, predict constantly within blocks, and flip their prediction between consecutive blocks. An example such expert ξ predicts

$$\xi = \underbrace{\mathsf{yyyyy}}_{\text{Block 1}}\overset{\text{Switch 1}}{\underset{\downarrow}{\mathsf{n}}}\underbrace{\mathsf{nnnnnn}}_{\text{Block 2}}\overset{\text{Switch 2}}{\underset{\downarrow}{\mathsf{n}}}\underbrace{\mathsf{yyyy}}_{\text{Block 3}}.$$

We may think of such switching experts either as static experts that switch between two *predictions* (c.f. Section 1.6) or equivalently as meta-experts that switch between the two constant *experts* (c.f. Section 1.9).

Figure 2.4 Backwards induction solution to the Ξ-regret game. For each sequence x of outcomes we have computed the value $\mathcal{V}_x$ of the subgame $\mathcal{G}_x$. Squares display the best expert(s) for x of length $t = T$. Ellipses give our minimax strategy $\hat{\sigma}_x$ for x of length $t < T$. Adversary's maximin strategy satisfies $\hat{\tau}_x = 1/2$ by Theorem 2.8.1.



The number of *blocks* always exceeds the number of *switches* by one, since switches occur between blocks. Whereas competing with the best constant expert allows us to learn the global trend of the sequence of outcomes, competing with switching experts allows us to track local trends. This is useful in applications where we expect the outcome-generating process to infrequently change, for example because it can be in several states. Our goal is to obtain an efficient minimax strategy.

The sets of prediction sequences of length T with *exactly* and *at most* m blocks are defined by

$$\mathbb{S}_T^m \coloneqq \left\{ \xi \in \{\mathsf{y}, \mathsf{n}\}^T \;\middle|\; \sum_{1 \leq t < T} \mathbf{1}_{\xi_t \neq \xi_{t+1}} = m - 1 \right\} \quad \text{and} \quad \mathbb{S}_T^{\leq m} \coloneqq \bigcup_{i \in [m]} \mathbb{S}_T^i.$$

Our example expert ξ above is a member of $\mathbb{S}_{16}^3$ and of $\mathbb{S}_{16}^{\leq m}$ for all $m \geq 3$. The numbers of such experts with exactly and at most m blocks equal

$$\left|\mathbb{S}_T^m\right| = 2\binom{T-1}{m-1} \qquad \text{and} \qquad \left|\mathbb{S}_T^{\leq m}\right| = 2\sum_{i \in [m]} \binom{T-1}{i-1}.$$

We now compete with the rather large set $\mathbb{S}_T^{\leq m}$. The cumulative loss of the best expert in this set is given by

$$L^*(x) \coloneqq \min_{\xi \in \mathbb{S}_T^{\leq m}} L(\xi, x)$$

The function L^* is obviously 1-Lipschitz, since the cumulative loss of *any* sequence of predictions changes by at most one if a single outcome is flipped, and hence so does the cumulative loss of the best expert in $\mathbb{S}_T^{\leq m}$.

The *switching regret game with time horizon* T is obtained by substituting this particular L^* into equation (2.7). The set $\mathbb{S}_T^{\leq m}$ is rather large, and hence computing the cumulative loss of the best expert by iterating over its members quickly becomes impractical, both in T and m. Fortunately, the set $\mathbb{S}_T^{\leq m}$ is highly regular, and its structure can be used to evaluate L^*, allowing us to efficiently execute our minimax strategy.

2.9.1 Executing Our Minimax Strategy

We saw in Section 2.8 that we can follow our minimax strategy once we have an efficient method to evaluate $L^*(x)$, the cumulative loss of the

best expert on outcomes x. In case of switching, the cumulative loss of the best expert can be computed efficiently by dynamic programming. The trick is to recursively define $L_{\mathsf{y}}^m(x)$, the cumulative loss on outcomes x of the *best* sequence of predictions with at most m blocks that predicts y next. The recursive clauses are, for $m > 1$

$$\begin{aligned} L_{\mathsf{y}}^m(x\mathsf{y}) &:= L_{\mathsf{y}}^m(x) & L_{\mathsf{y}}^m(x\mathsf{n}) &:= \min\bigl\{L_{\mathsf{y}}^m(x)+1, L_{\mathsf{n}}^{m-1}(x)\bigr\} \\ L_{\mathsf{n}}^m(x\mathsf{n}) &:= L_{\mathsf{n}}^m(x) & L_{\mathsf{n}}^m(x\mathsf{y}) &:= \min\bigl\{L_{\mathsf{n}}^m(x)+1, L_{\mathsf{y}}^{m-1}(x)\bigr\} \end{aligned}$$

and the base cases are

$$\begin{aligned} L_{\mathsf{y}}^m(\epsilon) &:= 0 & L_{\mathsf{y}}^1(x\mathsf{y}) &:= L_{\mathsf{y}}^1(x) & L_{\mathsf{y}}^1(x\mathsf{n}) &:= L_{\mathsf{y}}^1(x)+1 \\ L_{\mathsf{n}}^m(\epsilon) &:= 0 & L_{\mathsf{n}}^1(x\mathsf{n}) &:= L_{\mathsf{n}}^1(x) & L_{\mathsf{n}}^1(x\mathsf{y}) &:= L_{\mathsf{n}}^1(x)+1 \end{aligned}$$

An easy induction on both m and the length of x shows that

2.9.1. Proposition. *Fix $1 \leq m \leq T$. For each sequence x of T outcomes*

$$L^*(x) \;=\; \min\bigl\{L_y^m(x), L_n^m(x)\bigr\}.$$

Evaluating $L^*(x)$ for a history of length T can hence be done in time $O(mT)$. Predicting T outcomes using the minimax Algorithm 2.1 with this method of evaluating L^* takes $O(mT^2)$ time in total, and uses $O(m)$ memory.

2.9.2 Minimax Regret

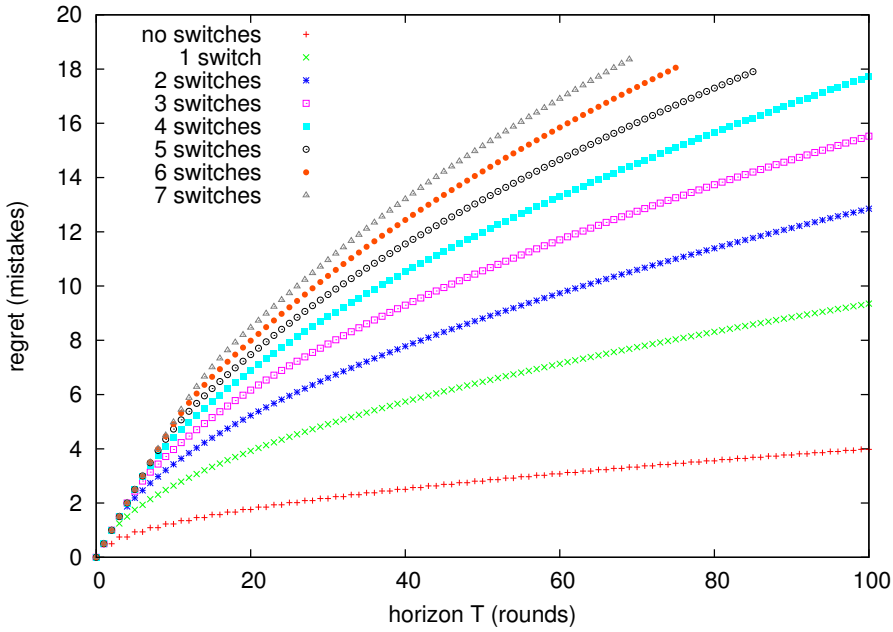
To evaluate our minimax regret, we use the fact that Adversary's maximin strategy, which generates outcomes uniformly at random, is an equaliser strategy. Our expected regret is thus

$$\mathcal{V} \;=\; T/2 - \mathbb{E}\bigl[L^*(x)\bigr]$$

irrespective of the strategy that we follow. The difficulty is, of course, to evaluate the expectation.

Using a trick similar to the dynamic programming solution of Section 2.9.1, we can evaluate our minimax regret $\mathcal{V}$ exactly for particular block counts m and time horizons T, resulting in Figure 2.5. For $m = 1$ blocks there are no switches, and our minimax regret is $\sqrt{T/(2\pi)}$ as before. We see that each additional block increases our minimax regret by less.

Figure 2.5 Minimax regret of the regret game with 0–5 switches as a function of time horizon T



Adversarial Upper Bound The power of Adversary increases when we give her $|\mathbb{S}_T^{\leq m}|$ many adversarial experts. Hence, our minimax regret is bounded by the fully adversarial bound of [25, Theorem 2.3], yielding

$$\sqrt{T/2 \ln |\mathbb{S}_T^{\leq m}|}$$

Since $|\mathbb{S}_T^m| \leq (T-1)^{m-1}$, we have

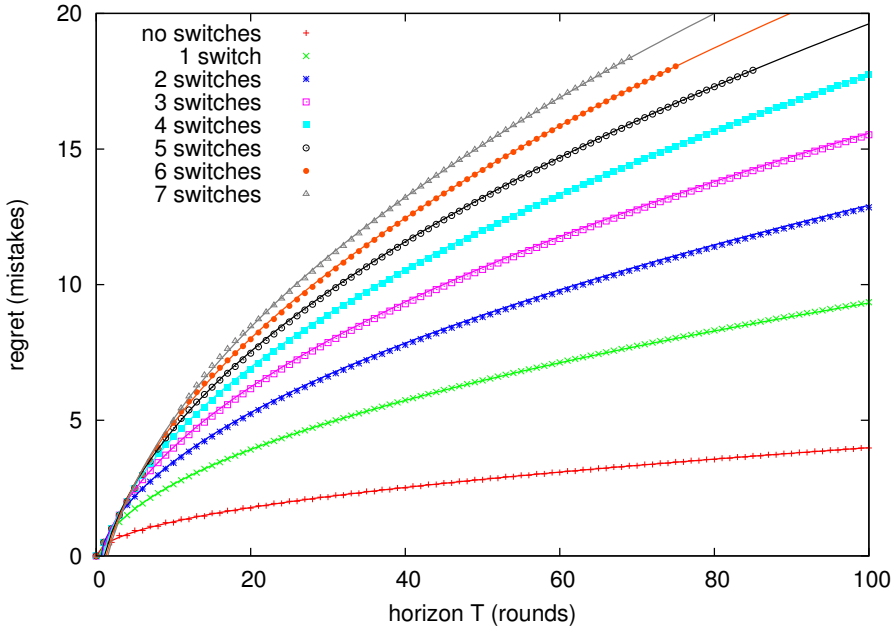
$$|\mathbb{S}_T^{\leq m}| \;\leq\; 2 \sum_{i=0}^{m-1} (T-1)^i \;=\; 2 \frac{(T-1)^m - 1}{(T-1) - 1} \;\leq\; 2T^{m-1}.$$

So that when $T \geq 2$, the minimax regret is bounded by

$$\sqrt{T/2 \ln |\mathbb{S}_T^{\leq m}|} \;\leq\; \sqrt{T/2m \ln(T)}. \tag{2.11}$$

By construction, this bound is too pessimistic. We now quantify the discrepancy.

Figure 2.6 Empirical Fit, equation (2.12), overlaid on Figure 2.5. For $m = 1$ (shown in red), we have graphed the minimax regret.



Empirical Fit Looking at Figure 2.5, it seems that the minimax regret is well approximated, for all $m \geq 2$, by

$$\sqrt{6(m-1)\frac{T}{2\pi}} - \frac{m-1}{\sqrt{5}}. \tag{2.12}$$

Figure 2.6 displays the fitted equation (2.12) overlaid on the exact minimax regrets from Figure 2.5. Indeed, we see that the fit is highly accurate. This suggests that (2.11) may be improved, getting rid of the logarithmic dependence on T under the square root.

We are in the curious situation that we have the optimal algorithm, but it is hard to quantify its guarantees. Deriving a better analytic bound is left as an open problem.

2.10 Related Research

We review the literature on online learning with the 0/1 loss.

Algorithm 2.2 Minimax algorithm for loss horizon k

Input: Game $\mathcal{G}_x$ with outcome sequence x.
Draw samples $z = z_1, z_2, \ldots$ uniformly at random from $\{\mathsf{y}, \mathsf{n}\}$ until the worst expert on xz dies, i.e. reaches cumulative loss $k+1$.
Then follow the other, live expert.

Prediction Problems The early work on sequential prediction with 0/1 loss includes the Weighted Majority algorithm [108] for combining the binary advice of n adversarial experts and the Aggregating algorithm [181]. These algorithms have a *learning rate* that needs to be supplied, and which must be tuned based on knowledge of time horizon T or loss horizon k. Uniform bounds can be obtained by the doubling trick, or more sophisticatedly, by decreasing the learning rate incrementally, as analysed e.g. by [25] for the exponentially weighted average forecaster.

A game-theoretic optimal solution for prediction with adversarial experts with loss horizon k was found much later in the form of the Binning algorithm by [4].

Decision Problems In prediction problems, experts suffer loss because they issue incorrect predictions. In the more abstract decision problems the predictions disappear, and experts are entities that can be followed and that suffer loss. Freund and Schapire adapted Weighted Majority to this setting, yielding the Hedge algorithm [59], and proved the regret bound we presented in Section 1.5.1 of the introduction Chapter 1.

The minimax algorithm for the decision problem with loss horizon k was recently obtained by Abernethy, Warmuth and Yellin in [6]. For comparison, we have displayed this algorithm, specialised to predicting two outcomes with our two constant experts, as Algorithm 2.2. This algorithm is also minimax for prediction with two constant experts, because two constant experts are essentially adversarial by the reasoning explained in Section 2.6.3. Both Algorithm 2.1 and Algorithm 2.2 sample a single sequence of random outcomes until the game is over. However, this outcome is used in a fundamentally different way. Algorithm 2.1 uses it to gauge the quality of both possible *predictions*. On the other hand, Algorithm 2.2 uses it estimate the eventual *best expert*.

Bandit Problems In the even more abstract bandit problems [10, 118, 176, 155, 5, 26, 24, 50, 55, 143, 158], the loss feedback disappears, and we only observe the loss of the expert we choose to follow, or of the action that we choose to play. This leads to the problem of *exploration vs exploitation*. To identify the best expert, we need to follow (exploration) all experts once in a while, to monitor their quality. But to suffer small loss, we want to follow (exploitation) the best expert exclusively. The challenge in bandit problems is to interleave these two goals, trading off the benefits of both.

Absolute Loss The *absolute loss* of a randomised prediction is defined as its expected $0/1$ loss [25]. Prediction with $0/1$ loss and with absolute loss are hence tightly related: their minimax regrets and strategies are identical, modulo one important and subtle difference. To issue minimax predictions under $0/1$ loss, it suffices to *randomly* compute $\hat{\sigma}$, the optimal probability of predicting y. For absolute loss it needs to be computed *exactly*. This means that Algorithm 2.1 cannot be used for absolute loss.

2.11 Conclusion

We presented and analysed four prediction games. We first considered predicting a single binary outcome with the help of two constant experts. We showed that the corresponding one-shot regret game has minimax regret $1/2$, which we can achieve by predicting uniformly at random. We then proceeded to predicting a sequence of binary outcomes. We first considered competing with the best constant expert for T trials. In the corresponding regret game with time horizon T our minimax regret essentially equals $\sqrt{T/(2\pi)}$. We then considered competing with the best constant expert given that the best expert makes at most k mistakes. In the regret game with loss horizon k, our minimax regret essentially equals $\sqrt{k/\pi}$. We concluded by competing with the best expert that switches at most $m-1$ times. In each case, we gave an efficient randomised algorithm for playing the minimax strategy.

The prediction problems that we considered form just the tip of the iceberg of online learning, and many interesting problems readily suggest themselves.

1. Prediction problems with more than two outcomes.
2. Decision problems (without outcomes).
3. More than two experts.
4. More complicated white-box experts.
5. Black-box (adversarial) experts.
6. More complicated (gray-box) meta-experts, e.g. that
   (a) switch between all of the above
   (b) combinatorially combine all of the above
7. Competing with gray-box experts with loss horizon k.
8. All the above for loss functions other than the 0/1 loss.

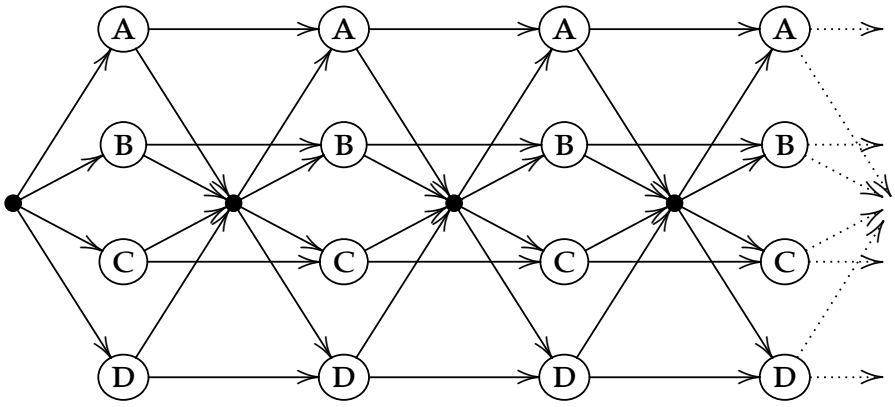
In the remainder of this dissertation we consider the following specific problems. In Chapter 3 we introduce a general graphical framework for the construction of meta-experts for log loss, with a focus on switching. In Chapter 4 we extend this framework to switching between gray-box experts. In particular, this allows us to switch between learning experts. Then in Chapter 5 we introduce a novel way to switch between two experts, motivated by finance but generally applicable to decision problems with log-loss or log-return. Finally, in Chapter 6 we consider combinatorial combinations of experts for 0/1 loss and/or absolute loss.

The approach taken in future chapters is different from the method developed in this chapter. The decision problems considered there are more complicated, and hence finding the minimax solutions is generally hard. Instead, we design strategies based on other motivations, and show that they guarantee small worst-case regret. In several cases we also prove lower bounds that show that these strategies are minimax up to constant factors.

For the problems that are not considered in this dissertation, we refer the interested reader to the standard textbook [25] as a starting point.

Chapter 3

Expert Hidden Markov Models



Abstract We show how models for prediction with expert advice can be defined concisely and clearly using hidden Markov models (HMMs); standard algorithms can then be used to efficiently calculate how the expert predictions should be weighted. We focus on algorithms for "tracking the best expert", starting from the fixed share algorithm, and show how most existing models can be cast as HMMs. We recover the running times and loss bounds for each algorithm, and discuss how they are related. We also describe three new models: (i) models with decreasing switching rate, which run in linear time and for which a fixed switching rate does not have to be specified in advance, (ii) a new generalisation of the fixed share algorithm that is especially well equipped to handle switches that occur in clusters, and (iii) a model tailored to the scenario where the experts have a natural order, and where jumps between them are typically small. This last model is relevant for predicting time series data where parameter drift is expected.

3.1 Introduction

We cannot predict exactly how complicated processes such as social interactions, the weather, the stock market and so on, will develop into the future. Nevertheless, people do make weather forecasts and buy shares all the time. Such predictions can be based on formal models, or on human expertise or intuition. An investment company may even want to choose between portfolios on the basis of a combination of these kinds of predictors. In such scenarios, it is typically unclear to what extent the predictors describe the true process underlying the data. Thus, we may well end up in a position where we have a whole collection of prediction strategies, or *experts*, each of whom has *some* insight into *some* aspects of the process of interest. We address the question how a given set of experts can be combined into a single predictive strategy that is as good as, or if possible even better than, the best individual expert.

The setup is as follows. Let Ξ be a set of experts. Each expert $\xi \in \Xi$ issues a distribution $P_\xi(\boldsymbol{x}_{t+1}|x^t)$ on the next outcome $\boldsymbol{x}_{t+1}$ given the previous observations $x^t := x_1, \ldots, x_t$. Here, each outcome x_i is an element of some countable space $\mathcal{X}$, and random variables are written in bold face. Then, when the outcome $\boldsymbol{x}_{t+1} = x_{t+1}$ is observed, the expert suffers *logarithmic loss* $-\ln P_\xi(x_{t+1}|x^t)$. Thus, if the experts assigns high probability to the actual observation, he incurs only small loss, and vice versa. When predicting a *sequence* of outcomes, the goal is to minimise the accumulated log-loss, but because of the chain rule of sequential probability, this amounts to the same thing as maximising the probability. For example, if we follow the predictions of expert ξ, our accumulated loss is

$$\sum_{i=1}^{t} -\ln P_\xi(x_i|x^{i-1}) = -\ln \prod_{i=1}^{t} \frac{P_\xi(x^i)}{P_\xi(x^{i-1})} = -\ln P_\xi(x^t).$$

Now that we have described how a single expert makes predictions and incurs loss, we must consider how such predictions can be combined into a single prediction strategy, such that the accumulated loss for this aggregate strategy is small. The standard Bayesian approach is to define a prior w on the experts Ξ which induces a joint distribution with mass function $P(x^t, \xi) = w(\xi)P_\xi(x^t)$. Inference is then based on this joint distribution. We can compute, for example: (a) the

marginal probability of the data $P(x^t) = \sum_{\xi \in \Xi} w(\xi) P_\xi(x^t)$, (b) the *predictive distribution* on the next outcome $P(\boldsymbol{x}_{t+1}|x^t) = P(x^t, \boldsymbol{x}_{t+1})/P(x^t)$, which defines a prediction strategy that combines those of the individual experts, or (c) the *posterior distribution* on the experts $P(\boldsymbol{\xi}|x^t) = P_{\boldsymbol{\xi}}(x^t)w(\boldsymbol{\xi})/P(x^t)$, which tells us how the experts' predictions should be weighted. This simple probabilistic approach has the advantage that it is computationally easy: predicting t outcomes using $k := |\Xi|$ experts requires only $O(k\,t)$ time. Additionally, this Bayesian strategy guarantees that the accumulated loss is only a constant $-\ln w(\hat{\xi})$ smaller than the loss incurred by best available expert $\hat{\xi}$. On the flip side, with this strategy we never do any *better* than $\hat{\xi}$ either: we have $-\ln P_{\hat{\xi}}(x^t) \leq -\ln P(x^t) \leq -\ln P_{\hat{\xi}}(x^t) - \ln w(\hat{\xi})$, which means that potentially valuable insights from the other experts are not used to our advantage!

More sophisticated methods to combine prediction strategies can be found in the literature under various headings. On the one hand there are results in (Bayesian) statistics and source coding. On the other hand, the learning theory community has produced a lot of work on universal prediction under the heading "prediction with expert advice". In this case the experts' predictions are not necessarily probabilistic, and scored using an arbitrary loss function. In this chapter we state our results for logarithmic loss. Our results apply to any mixable loss function, as discussed in Section 3.6.4.

The three main contributions of this chapter are the following. First, we introduce prior distributions on *sequences* of experts, which allows unified description of many existing models. Second, we show how HMMs can be used as an intuitive graphical language to describe such priors and obtain computationally efficient prediction strategies. Third, we use this new approach to describe and analyse numerous models for *expert tracking*. On the one hand, we summarise some of the most influential existing results in this area, while on the other hand we introduce a number of new models that represent new good trade-offs between time complexity and modelling power.

3.1.1 Overview

In Section 3.2 we develop a new, more general framework for combining expert predictions, where we consider the possibility that the

optimal weights used to mix the expert predictions may *vary over time*, i.e. as the sample size increases. We stick to Bayesian methodology, but we define the prior distribution as a probability measure on *sequences of experts* rather than on experts. The prior probability of a sequence $\xi_1, \xi_2, \ldots$ is the probability that we rely on expert ξ_1's prediction of the first outcome and expert ξ_2's prediction of the second outcome, etc. To see why this is useful, consider that the nature of the data generating process may evolve over time; consequently different experts may be better during different periods of time. It is also possible that not the data generating process, but the experts themselves change as more and more outcomes are being observed: they may learn from past mistakes, possibly at different rates, or they may have occasional bad days, etc. In both situations we may hope to benefit from more sophisticated modelling.

Of course, not all models for combining expert predictions are computationally feasible. Section 3.3 describes a methodology for the specification of models that allows efficient evaluation. We achieve this by using hidden Markov models (HMMs) on two levels. On the first level, we use an HMM to specify a distribution on sequences of *experts* as defined in Section 3.2. We introduce a graphical language to conveniently represent its structure. These graphs help to understand and compare existing models and to design new ones. We then modify this first HMM to construct a second HMM that specifies the distribution on sequences of *outcomes*. Subsequently, we can use the standard dynamic programming algorithms for HMMs (forward, backward and Viterbi) on both levels to efficiently calculate most relevant quantities, most importantly the marginal probability of the observed outcomes $P(x^t)$ and posterior weights on the next expert given the previous observations $P(\boldsymbol{\xi}_{t+1}|x^t)$.

Many existing models for prediction with expert advice can be specified by HMMs that in turn define expert sequence priors (ES-priors). We are interested in evaluating these models in terms of two qualities: on the one hand, we want to know the time and space complexities of predicting t outcomes using k experts. On the other hand, we evaluate the predictive performance of the models by giving *regret bounds*. The *regret* is the discrepancy between the predictive performance of the considered model, and the performance of another prediction strategy from a set of reference strategies. While the time complexity of any

model can be read directly from the structure of its defining HMM, we need some theory that is developed in Section 3.4 to assist in analysing the regret.

We proceed in Section 3.5 with a discussion of numerous models for tracking the best expert, where the *fixed share* algorithm [79, 80] serves as a starting point. We identify two drawbacks of fixed share: first, one has to specify a fixed *switching rate* in advance; choosing a suboptimal value here produces a linear penalty in the regret. Second, the incurred regret depends on the sample size t, even when the optimal number of switches is bounded. These problems can be addressed by modelling the switching probabilities differently. Section 3.5.2 explains how the part of the model that describes switching probabilities can be isolated from the rest; we then proceed to describe several alternative models for the switching probabilities, and discuss how these modifications affect the regret bound. In particular, Section 3.5.3.2 describes a new, simple and effective approach to solve both problems associated with fixed share, and Section 3.5.5 also describes a new model that is especially well suited to the scenario where changes in predictive performance are expected to appear in clusters.

So far, none of the considered models for expert tracking made any assumptions as to the inner workings of, or the relationships between, the various experts – they are black boxes. However, as an interesting special case we also consider the scenario where the experts are ordered. For example, if the experts are prediction strategies associated with a parametric model, instantiated with various parameter values, then switches between two experts seem intuitively more likely if they represent parameter values that are close. This scenario is explored in Section 3.5.6; the notion is taken to its extreme in Section 3.5.7, where the regret is no longer analysed in terms of all-or-nothing "switches", but rather in terms of a more smooth characterisation of the amount of "parameter drift".

A number of loose ends associated with prediction with expert advice and expert tracking based on HMMs and ES-priors are discussed in Section 3.6. We specifically discuss an extension of our framework that generalises the concept of ES-priors by allowing the probability of the next expert to depend on the observed data (Section 3.6.1), how one might estimate which expert made the best prediction at a certain time (Section 3.6.2), and finally how the framework we consider actually ap-

plies generally to any mixable loss function (Section 3.6.4).

3.2 Expert Sequence Priors

In this section we explain how expert tracking can be described in probability theory using expert sequence priors. These ES-priors are distributions on the space of infinite sequences of experts that are used to express regularities in the development of the relative quality of the experts' predictions. As illustrations we render Bayesian mixtures and elementwise mixtures as ES-priors. In the next section we show how ES-priors can be implemented efficiently by hidden Markov models.

Notation We denote by $\mathbb{N}$ the natural numbers including zero, and by $\mathbb{Z}_+$ the natural numbers excluding zero. Let Q be a set. We denote the cardinality of Q by $|Q|$. For any natural number t, we let the variable q^t range over the t-fold Cartesian product Q^t, and we write $q^t = \langle q_1, \ldots, q_t \rangle$. We also let q^ω range over Q^ω — the set of infinite sequences over Q — and write $q^\omega = \langle q_1, \ldots \rangle$. We read the statement $q^\lambda \in Q^{\leq \omega}$ to first bind $\lambda \leq \omega$ and subsequently $q^\lambda \in Q^\lambda$. If q^λ is a sequence, and $\kappa \leq \lambda$, then we denote by q^κ the prefix of q^λ of length κ.

Forecasting System Let $\mathcal{X}$ be a countable outcome space. We use the notation $\mathcal{X}^*$ for the set of all finite sequences over $\mathcal{X}$ and let $\mathrm{prob}(\mathcal{X})$ denote the set of all probability mass functions on $\mathcal{X}$. A *(prequential) $\mathcal{X}$-forecasting system* (PFS) is a function $P : \mathcal{X}^* \to \mathrm{prob}(\mathcal{X})$ that maps sequences of previous observations to a predictive distribution on the next outcome. Prequential forecasting systems were introduced in [39].

Distributions We also use probability measures on spaces of infinite sequences. In such a space, a basic event is the set of all continuations of a given prefix. We identify such events with their prefix. Thus a distribution on $\mathcal{X}^\omega$ is defined by a function $P : \mathcal{X}^* \to [0, 1]$ that satisfies $P(\epsilon) = 1$, where ϵ is the empty sequence, and for all $t \geq 0$, all $x^t \in \mathcal{X}^t$ we have $\sum_{x \in \mathcal{X}} P(x_1, \ldots, x_t, x) = P(x^t)$. We identify P with the distribution it defines. We write $P(x^t | x^i)$ for $P(x^t)/P(x^i)$ if $0 \leq i \leq t$.

Note that forecasting systems continue to make predictions even after they have assigned probability 0 to a previous outcome, while

distributions' predictions become undefined. Nonetheless we use the same notation: we write $P(x_{t+1}|x^t)$ for the probability that a forecasting system P assigns to the next outcome given the first t outcomes, as if P were a distribution.

ES-Priors The slogan of this chapter is *we do not understand the data*. Instead of modelling the data, we work with experts. We assume that there is a fixed set of k experts Ξ, and that each expert $\xi \in \Xi$ predicts using a forecasting system P_ξ.

We are interested in switching between different forecasting systems (experts) at different sample sizes. For a sequence of experts with prefix ξ^t, the combined forecast, where expert ξ_i predicts the ith outcome, is denoted

$$P_{\xi^t}(x^t) \;:=\; \prod_{i=1}^{t} P_{\xi_i}(x_i|x^{i-1}).$$

Adopting Bayesian methodology, we impose a prior π on infinite sequences of experts; this prior is called an *expert sequence prior* (ES-prior). Inference is then based on the distribution on the joint space $(\mathcal{X} \times \Xi)^\omega$, called the *ES-joint*, which is defined as follows:

$$P_\pi(\xi^t, x^t) \;:=\; \pi(\xi^t) P_{\xi^t}(x^t). \tag{3.1}$$

For example, this ES-joint induces a marginal distribution on sequences of outcomes:

$$P_\pi(x^t) = \sum_{\xi^t \in \Xi^t} P_\pi(\xi^t, x^t). \tag{3.2}$$

Compare this to the usual Bayesian statistics, where a model class $\{P_\theta \mid \theta \in \Theta\}$ is also endowed with a prior distribution w on Θ. Then, after observing outcomes x^t, inference is based on the posterior distribution on the parameter $\boldsymbol{\theta} \in \Theta$, which is never actually observed. Our approach is exactly the same, but we always consider $\Theta = \Xi^\omega$. Thus as usual predictions are based on the posterior distribution on $\boldsymbol{\xi}^\omega$. Namely, at each moment in time the predictions of the experts are weighted according to the posterior probability of $\boldsymbol{\xi}_{t+1}$, which can be computed as follows:

$$P_\pi(\boldsymbol{\xi}_{t+1}|x^t) = \frac{\sum_{\xi^t} P_\pi(\xi^t, x^t, \boldsymbol{\xi}_{t+1})}{\sum_{\xi^t} P_\pi(\xi^t, x^t)} = \frac{\sum_{\xi^t} \pi(\xi^t, \boldsymbol{\xi}_{t+1}) P_{\xi^t}(x^t)}{\sum_{\xi^t} \pi(\xi^t) P_{\xi^t}(x^t)}. \tag{3.3}$$

Note that although this probability can be evaluated in principle, it involves summing an exponential number of terms in general, which is hardly practical. In the subsequent sections of this chapter we will be looking for ES-priors that allow for efficient evaluation.

In the traditional subjectivist Bayesian interpretation, the prior distribution expresses our beliefs about an unknown "true" parameter value, but in the case of ES-priors this philosophy is tenuous: normally there is no "true expert sequence", as experts do not generate data, they only predict it. Moreover, by mixing different expert sequences, it is sometimes possible to predict significantly better than by using any single sequence of experts. Ideally, the ES-prior π should be chosen such that the posterior distribution on experts (3.3) coincides with the optimal mixture weights.

Rather than appealing to a subjectivist philosophy, in the remainder of this chapter we motivate ES-priors by giving performance guarantees in the form of bounds on running time and regret.

3.2.1. Example (Bayesian Mixtures). Let Ξ be a set of experts, and let P_ξ be a PFS for each $\xi \in \Xi$. Suppose that we do not know which expert will make the best predictions. Following the usual Bayesian methodology, we combine their predictions by conceiving a prior w on Ξ, which (depending on the adhered philosophy) may or may not be interpreted as an expression of beliefs in this respect. Then the standard Bayesian mixture $P_{\text{bayes},w}$ is given by

$$P_{\text{bayes},w}(x^t) := \sum_{\xi \in \Xi} P_\xi(x^t) w(\xi). \tag{3.4}$$

Recall that $P_\xi(x^t)$ means $\prod_{i=1}^t P_\xi(x_i|x^{i-1})$, the combined prediction of expert ξ. The Bayesian mixture is not an ES-joint, but it can easily be transformed into one by using the ES-prior that assigns probability $w(\xi)$ to the identically-ξ sequence for each $\xi \in \Xi$:

$$\pi_{\text{bayes},w}(\xi^t) := \begin{cases} w(k) & \text{if } \xi_i = k \text{ for all } i = 1, \ldots, t, \\ 0 & \text{o.w.} \end{cases}$$

We will use the adjective "Bayesian" generously throughout this chapter, but when we write *the standard Bayesian ES-prior* this always refers to $\pi_{\text{bayes},w}$. ◇

3.2.2. Example (Elementwise Mixtures). Again fix experts Ξ. The *elementwise mixture*[1] is formed from some mixture weights $w \in \text{prob}(\Xi)$ by

$$P_{\text{mix},w}(x^t) \;:=\; \prod_{i=1}^{t} \sum_{\xi \in \Xi} P_\xi(x_i|x^{i-1}) w(\xi).$$

It may seem that elementwise mixtures do not fit in the framework of ES-priors. But we can rewrite this definition in the required form as follows:

$$P_{\text{mix},w}(x^t) \;=\; \prod_{i=1}^{t} \sum_{\xi \in \Xi} P_\xi(x_i|x^{i-1}) w(\xi) \;=$$

$$\sum_{\xi^t \in \Xi^t} \prod_{i=1}^{t} P_{\xi_i}(x_i|x^{i-1}) w(\xi_i) \;=\; \sum_{\xi^t} P_{\xi^t}(x^t) \pi_{\text{mix},w}(\xi^t),$$

which is the ES-joint based on the ES-prior

$$\pi_{\text{mix},w}(\xi^t) \;:=\; \prod_{i=1}^{t} w(\xi_i). \tag{3.5}$$

Thus, the ES-prior for elementwise mixtures is just the product distribution of w. ◇

We warned earlier against the interpretation of ES-priors as expressions of belief about individual expert sequences. This is an example where the ES-prior is clearly used differently: it is crafted such that its posterior $\pi_{\text{mix},w}(\boldsymbol{\xi}_{t+1}|\xi^t) = w(\boldsymbol{\xi}_{t+1})$ exactly coincides with the desired *mixture* of experts.

3.3 Expert Tracking using HMMs

We explained in the previous section how expert tracking can be implemented using expert sequence priors. In this section we specify ES-priors using hidden Markov models (HMMs). The advantage of using HMMs is twofold. HMM state transition diagrams clearly and

[1] These mixtures are sometimes just called mixtures, or predictive mixtures, or exponentially weighted averages. We use the term elementwise mixtures both for descriptive clarity and to avoid confusion with Bayesian mixtures.

intuitively display the model structure. Moreover, the time complexity of the resulting expert tracking procedure can be read off directly from these diagrams. We first give a short overview of the particular kind of HMMs that we use throughout this chapter. We then show how HMMs can be used to specify ES-priors. As illustrations we render the ES-priors that we obtained for Bayesian mixtures and elementwise mixtures in the previous sections, as HMMs.

3.3.1 Hidden Markov Models Overview

Hidden Markov models (HMMs) are a well-known tool for specifying probability distributions on sequences with temporal structure, in this case expert sequences. Furthermore, these distributions are very appealing algorithmically: many important probabilities can be computed efficiently for HMMs. These properties make HMMs ideal models for the definition of ES-priors. For an introduction to HMMs, see [146]. We require a slightly more general notion that incorporates silent states and forecasting systems as explained below.

We define our HMMs on a generic set of outcomes $\mathcal{O}$ to avoid confusion in later sections, where we use HMMs in three different contexts. First in Section 3.3.2, we use HMMs to define ES-priors, and instantiate $\mathcal{O}$ with the set of experts Ξ. Then in Section 3.3.3 we modify the HMM that defines the ES-prior to incorporate the experts' predictions, whereupon $\mathcal{O}$ is instantiated with the set of observable outcomes $\mathcal{X}$. Finally, in Section 3.5.2 we build HMMs that output (binary) switch decisions, and we set $\mathcal{O} = \{\mathsf{n}, \mathsf{s}\}$.

3.3.1. Definition. Let $\mathcal{O}$ be a finite set of outcomes. We call a quintuple

$$\mathfrak{H} = \left\langle Q, Q_{\mathrm{p}}, \mathrm{P}_{\circ}, \mathrm{P}_{\rightarrow}, \left\langle \mathrm{P}_{\downarrow q} \right\rangle_{q \in Q_{\mathrm{p}}} \right\rangle$$

a *hidden Markov model* on $\mathcal{O}$ if Q is a countable set, $Q_{\mathrm{p}} \subseteq Q$, $\mathrm{P}_{\circ} \in \mathrm{prob}(Q)$, $\mathrm{P}_{\rightarrow} : Q \to \mathrm{prob}(Q)$ and $\mathrm{P}_{\downarrow q}$ is an $\mathcal{O}$-forecasting system for each $q \in Q_{\mathrm{p}}$.

Terminology and Notation We call elements of Q *states*. We call the states in Q_{p} *productive* and the other states *silent*. We call $\mathrm{P}_{\circ}$ the *initial distribution*, let I denote its support (i.e. $I := \{q \in Q \mid \mathrm{P}_{\circ}(q) > 0\}$) and call I the set of *initial states*. We call $\mathrm{P}_{\rightarrow}$ the *stochastic transition function*.

We let S_q denote the support of $\mathrm{P}_{\rightarrow}(q)$, and call $q' \in S_q$ a *direct successor* of q. We abbreviate $\mathrm{P}_{\rightarrow}(q)(q')$ to $\mathrm{P}(q \to q')$. A finite or infinite sequence of states $q^\lambda \in Q^{\leq\omega}$ is called a *branch* through $\mathfrak{H}$. A branch q^λ is called a *run* if either $\lambda = 0$ (so $q^\lambda = \epsilon$), or $q_1 \in I$ and $q_{i+1} \in S_{q_i}$ for all $1 \leq i < \lambda$. A finite run $q^t \neq \epsilon$ is called *a run to* q_t. For each branch q^λ, we denote by q^λ_{p} its subsequence of productive states. We denote the elements of q^λ_{p} by q^{p}_1, q^{p}_2 etc. We call an HMM *continuous* if q^ω_{p} is infinite for each infinite run q^ω.

Restriction In this chapter we will only work with continuous HMMs. This restriction is necessary for the following to be well-defined.

3.3.2. Definition. An HMM $\mathfrak{H}$ defines the following distribution on sequences of states. $\pi_{\mathfrak{H}}(\epsilon) := 1$, and for $\lambda \geq 1$

$$\pi_{\mathfrak{H}}(q^\lambda) := \mathrm{P}_{\circ}(q_1) \prod_{i=1}^{\lambda-1} \mathrm{P}(q_i \to q_{i+1}).$$

Then via the PFSs, $\mathfrak{H}$ induces the joint distribution $P_{\mathfrak{H}}$ on runs and sequences of outcomes. Let $o^t \in \mathcal{O}^t$ be a sequence of outcomes and let q^λ be a run with at least t productive states, then

$$P_{\mathfrak{H}}(o^t, q^\lambda) := \pi_{\mathfrak{H}}(q^\lambda) \prod_{i=1}^{t} \mathrm{P}_{\downarrow q^{\mathrm{p}}_i}(o_i | o^{i-1}).$$

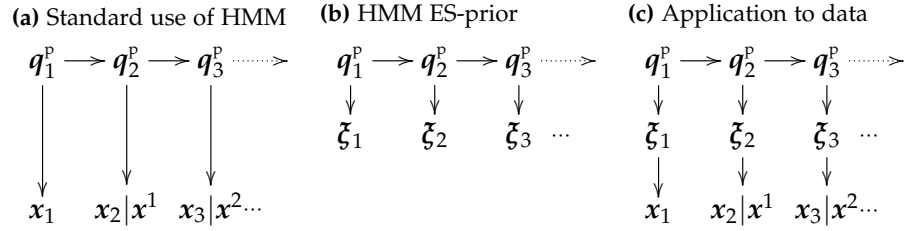
The value of $P_{\mathfrak{H}}$ at arguments o^t, q^λ that do not fulfil the condition above is determined by the additivity axiom of probability.

3.3.2 HMMs as ES-Priors

In the most straightforward application of HMMs, the hidden state determines a distribution on the observable outcomes. A graphical model depicting this approach is displayed in Figure 3.1a. However, in this chapter we use HMMs as ES-priors, that is, to specify temporal correlations between the performance of our *experts*. Thus instead of concrete observations our HMMs will "produce" sequences of experts, that are never actually observed. Figure 3.1b illustrates this.

Using HMMs as priors allows us to use the standard algorithms for HMMs to answer questions about the prior. For example, we can use

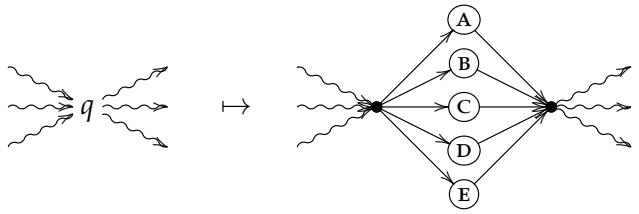
Figure 3.1 HMMs. q_i^{P}, ξ_i and x_i are the ith productive state, expert and observation.



the forward algorithm to compute the prior probability of the sequence of one hundred experts with expert number one at all odd indices and expert number two at all even indices. However, we are obviously also interested in questions about the data rather than about the prior. In Section 3.3.3 we show how joints based on HMM priors (Figure 3.1c) can be transformed into ordinary HMMs (Figure 3.1a) with at most a k-fold increase in size, allowing us to use the standard algorithms for HMMs not only for the experts, but for the data as well, with the same increase in complexity. This is the best we can generally hope for, as we now need to integrate over all possible expert sequences instead of considering only a single one. Here we first consider properties of HMMs that represent ES-priors.

Restriction HMM priors "generate", or define the distribution on, sequences of experts. But contrary to the data, which are observed, no concrete sequence of experts is realised. This means that we cannot conveniently condition the distribution on experts in a productive state q_t^{P} on the sequence of previously produced experts ξ^{t-1}. We therefore require that the forecasting systems associated to the states are fixed distributions, so that all dependencies between consecutive experts are carried by the state, in order to avoid having to sum over all (exponentially many) possible expert sequences. It is possible to relax this restriction somewhat: while it is not practical to condition on the sequence of previously produced experts, we can condition on (a function of) the observed data. This extension is discussed further in Section 3.6.1.

Deterministic Under the restriction above, and in the presence of silent states, we can make any HMM deterministic in the sense that the forecasting systems associated with the productive states assign probability one to a single outcome. We simply replace each productive state $q \in Q_{\mathrm{p}}$ by the following gadget:



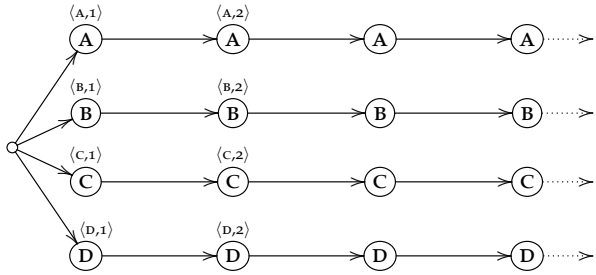
In the left diagram, the state q has distribution $\mathrm{P}_{\downarrow q}$ on outcomes $\mathcal{O} = \{\textsc{a}, \ldots, \textsc{e}\}$. In the right diagram, the leftmost silent state has transition probability $\mathrm{P}_{\downarrow q}(o)$ to a state that deterministically outputs outcome o. We call a HMM $\langle Q, Q_{\mathrm{p}}, \mathrm{P}_{\circ}, \mathrm{P}_{\rightarrow}, \Lambda \rangle$ on $\mathcal{O}$ *deterministic* if Λ maps productive states to fixed outcomes, i.e. it is a function $\Lambda : Q_{\mathrm{p}} \to \mathcal{O}$. This is a slight abuse of notation as the last component of a (general) HMM assigns a *PFS* to each productive state, while the last component of a deterministic HMM assigns an *outcome* to each productive state.

Sequential prediction using a general HMM or its deterministic counterpart costs the same amount of work: the $|\mathcal{O}|$-fold increase in the number of states is compensated by the $|\mathcal{O}|$-fold reduction in the number of outcomes that need to be considered per state. However, deterministic HMMs more accurately describe the operations of the forward algorithm, which now has to perform a constant amount of work for each edge (see Section 3.3.4.3), and they are convenient in diagrams.

Diagrams We graphically represent deterministic HMMs by drawing a node N_q for each state q. We draw silent states as small black dots, e.g. $\bullet$. We draw each productive state q as an open circle labelled by the produced expert $\Lambda(q)$, e.g. Ⓐ. We draw an arrow from N_q to $N_{q'}$ if q' is a direct successor of q. We often reify the initial distribution $\mathrm{P}_{\circ}$ by including a virtual node, drawn as an open circle, e.g. $\circ$, with an outgoing arrow to N_q for each initial state $q \in I$. The transition probability $P(q \to q')$ is not displayed in the graph.

We are now ready to give the deterministic HMMs that correspond

Figure 3.2 Standard Bayesian mixture $\textsc{bayes}[\Xi, w]$



$$\textsc{bayes}[\Xi, w] = \langle Q, Q_{\mathrm{p}}, \mathrm{P}_\circ, \mathrm{P}_\to, \Lambda \rangle$$
$$Q = Q_{\mathrm{p}} = \Xi \times \mathbb{Z}_+$$
$$\Lambda(\xi, t) = \xi \qquad \mathrm{P}_\circ(\xi, 1) = w(\xi)$$
$$\mathrm{P}(\langle \xi, t \rangle \to \langle \xi, t+1 \rangle) = 1$$

to the ES-priors of our running examples: Bayesian mixtures and elementwise mixtures with fixed parameters.

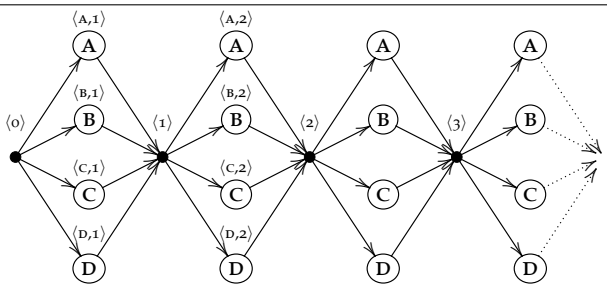
3.3.3. Example (HMM for Bayesian Mixtures). The Bayesian mixture ES-prior $\pi_{\text{bayes},w}$ as introduced in Example 3.2.1 represents the hypothesis that a single expert predicts best for all sample sizes. A simple deterministic HMM on Ξ that generates the prior $\pi_{\text{bayes},w}$ is given by $\textsc{bayes}[\Xi, w]$ as defined in Figure 3.2. Since the HMM computes the Bayesian mixture, (3.4) tells us that the regret (log-loss overhead) of $\textsc{bayes}[\Xi, w]$ w.r.t. each expert $\xi \in \Xi$ is bounded for all data x^t by

$$\ln \frac{P_\xi(x^t)}{P_{\textsc{bayes}[\Xi,w]}(x^t)} \;\leq\; -\ln w(\xi). \tag{3.6}$$

In particular this bound holds for $\hat{\xi} = \operatorname{argmax}_\xi P_\xi(x^t)$, so we predict as well as the single best expert with *constant* overhead. Also $P_{\text{bayes},w}(x^t)$ can obviously be computed in $O(k\,t)$ using its definition (3.4). We show in Section 3.3.4 that computing $P_{\textsc{bayes}[\Xi,w]}(x^t)$ has exactly the same running time. ◇

3.3.4. Example (HMM for Elementwise Mixtures). The deterministic HMM $\textsc{em}[\Xi, w]$ that implements the ES-prior $\pi_{\text{mix},w}$ of Example 3.2.2 is defined in Figure 3.3. The vector-style definition of $\mathrm{P}_\to$ is shorthand

Figure 3.3 Fixed elementwise mixture $\textsc{em}[\Xi, w]$



$$\textsc{em}[\Xi, w] \;=\; \langle Q, Q_{\mathrm{p}}, \mathrm{P}_{\circ}, \mathrm{P}_{\rightarrow}, \Lambda \rangle$$
$$Q = Q_{\mathrm{s}} \cup Q_{\mathrm{p}} \quad Q_{\mathrm{s}} = \mathbb{N} \quad Q_{\mathrm{p}} = \Xi \times \mathbb{Z}_{+}$$
$$\mathrm{P}_{\circ}(0) = 1 \qquad \Lambda(\xi, t) = \xi$$
$$\mathrm{P}\begin{pmatrix} \langle t \rangle \to \langle \xi, t+1 \rangle \\ \langle \xi, t \rangle \to \langle t \rangle \end{pmatrix} = \begin{pmatrix} w(\xi) \\ 1 \end{pmatrix}$$

for one $\mathrm{P}_{\rightarrow}$ per line. The HMM has a single silent state per outcome, whose transition probabilities are the mixture weights w. We show in Section 3.3.4 that this HMM allows $P_{\textsc{em}[\Xi,w]}(x^t)$ to be calculated in $O(k\,t)$ time as well. ◇

3.3.3 The HMM for Data

After composing an HMM prior on Ξ, we obtain our model for the data (Figure 3.1c) by introducing a PFS P_ξ for each expert $\xi \in \Xi$. As it turns out, the resulting marginal distribution on data can be implemented by a single HMM on $\mathcal{X}$ (Figure 3.1a) *with the same number of states as the HMM prior*. Let P_ξ be an $\mathcal{X}$-forecasting system for each $\xi \in \Xi$, and let the ES-prior $\pi_{\mathfrak{H}}$ be given by the deterministic HMM $\mathfrak{H} = \langle Q, Q_{\mathrm{p}}, \mathrm{P}_{\circ}, \mathrm{P}_{\rightarrow}, \Lambda \rangle$ on Ξ. Then the marginal distribution of the data (see (3.2)) is given by

$$P_{\mathfrak{H}}(x^t) = \sum_{\xi^t} \pi_{\mathfrak{H}}(\xi^t) \prod_{i=1}^{t} P_{\xi_i}(x_i | x^{i-1}).$$

The HMM $\mathbb{X} := \langle Q, Q_{\mathrm{p}}, \mathrm{P}_{\circ}, \mathrm{P}_{\rightarrow}, \langle P_{\Lambda(q)} \rangle_{q \in Q_{\mathrm{p}}} \rangle$ on $\mathcal{X}$ induces the same marginal distribution (see Definition 3.3.2). That is, $P_{\mathbb{X}}(x^t) = P_{\mathfrak{H}}(x^t)$. Moreover, $\mathbb{X}$ contains only the forecasting systems that also exist in $\mathfrak{H}$ and it retains the structure of $\mathfrak{H}$. In particular this means that the algorithms for HMMs have the *same* running time on the prior $\mathfrak{H}$ as on the marginal $\mathbb{X}$.

3.3.4 Computation

In this section we briefly review the important computational tasks and corresponding algorithms for HMMs. We then give the forward algorithm for our particular kind of HMMs with silent states, prove its correctness and bound its running time as a function of the input HMM and the sample size t.

3.3.4.1 Tasks & Algorithms

There are three tasks traditionally associated with hidden Markov models [146]:

1. Computing the marginal probability $P(x^t)$ of the data x^t and/or sequentially computing the prediction $P(\boldsymbol{x}_{t+1}|x^t)$ of the next outcome given past data. This task is performed by the *forward algorithm*, a dynamic programming algorithm that operates by percolating weights along the transitions of the HMM.

2. MAP calculation: computing a sequence of states q^λ with maximal posterior weight $P(q^\lambda|x^t)$. Note that $\lambda \geq t$. This task is solved using the *Viterbi* algorithm, a simple variation on the forward algorithm.

3. Parameter estimation. Instead of a single probabilistic transition function $\mathrm{P}_{\rightarrow}$, one may consider a collection of transition functions $\langle \mathrm{P}_{\rightarrow\theta} \mid \theta \in \Theta \rangle$ indexed by a set of parameters Θ. The *Baum-Welch* algorithm can then be used to find the parameter θ for which the corresponding HMM achieves the highest likelihood of the data. This is an iterative improvement algorithm (in fact an instance of Expectation Maximisation (EM)) built atop the forward algorithm and a related dynamic programming algorithm called the backward algorithm.

This chapter is mainly concerned with sequential prediction problems. Section 3.6.2 provides a short discussion of the intricacies of expert estimation: the problem of finding out which expert made the best predictions at which time step. Parameter estimation is outside the scope of this study.

We first describe the preprocessing step called *unfolding* and other preliminaries. We then describe a version of the forward algorithm that can cope with silent states and the additional layer that is introduced by the experts. We prove correctness and analyse its running time and space requirement.

3.3.4.2 Preliminaries

Unfolding For simplicity we will restrict attention to HMMs that are unfolded in the following sense. Every HMM can be transformed into an equivalent HMM in which each productive state is involved in the production of a unique outcome. For example, the single node in Figure 3.4a is involved in the production of $x_1, x_2, \ldots$ In its unfolding Figure 3.4b the i^{th} node is only involved in producing x_i. Figures 3.4c and 3.4d show HMMs that unfold to the Bayesian mixture shown in Figure 3.2 and the elementwise mixture shown in Figure 3.3. In full generality, fix an HMM $\mathfrak{H}$. Without loss of generality, assume that $\mathrm{P}_\circ$ is on productive states. The *unfolding* of $\mathfrak{H}$ is the HMM

$$\mathcal{U}(\mathfrak{H}) := \left\langle Q^{\mathrm{u}}, Q^{\mathrm{u}}_{\mathrm{p}}, \mathrm{P}^{\mathrm{u}}_\circ, \mathrm{P}^{\mathrm{u}}_\to, \left\langle \mathrm{P}^{\mathrm{u}}_{\downarrow q} \right\rangle_{q \in Q^{\mathrm{u}}} \right\rangle,$$

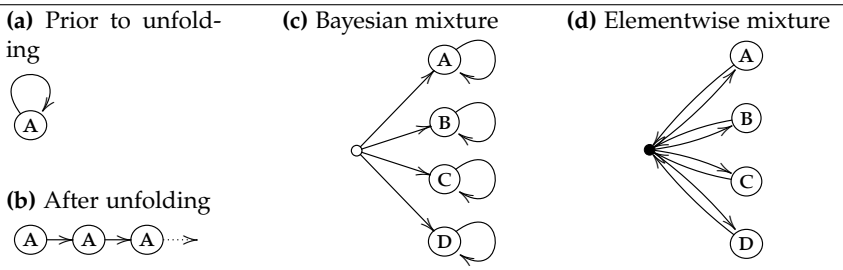
where the states and productive states are given by:

$$\begin{aligned} Q^{\mathrm{u}} &:= \left\{ \langle q_\lambda, t \rangle \mid q^\lambda \text{ is a run through } \mathfrak{H} \right\}, \quad \text{where } t = |q^\lambda_{\mathrm{p}}| \\ Q^{\mathrm{u}}_{\mathrm{p}} &:= Q^{\mathrm{u}} \cap (Q_{\mathrm{p}} \times \mathbb{N}) \end{aligned}$$

and the initial probability, transition function and forecasting systems are:

$$\begin{aligned} \mathrm{P}^{\mathrm{u}}_\circ\left(\langle q, 1 \rangle\right) &:= \mathrm{P}_\circ(q) \\ \mathrm{P}^{\mathrm{u}}\begin{pmatrix} \langle q, t \rangle \to \langle q', t+1 \rangle \\ \langle q, t \rangle \to \langle q', t \rangle \end{pmatrix} &:= \begin{pmatrix} \mathrm{P}(q \to q') \\ \mathrm{P}(q \to q') \end{pmatrix} \\ \mathrm{P}^{\mathrm{u}}_{\downarrow \langle q, t \rangle} &:= \mathrm{P}_{\downarrow q}. \end{aligned}$$

Figure 3.4 Unfolding example



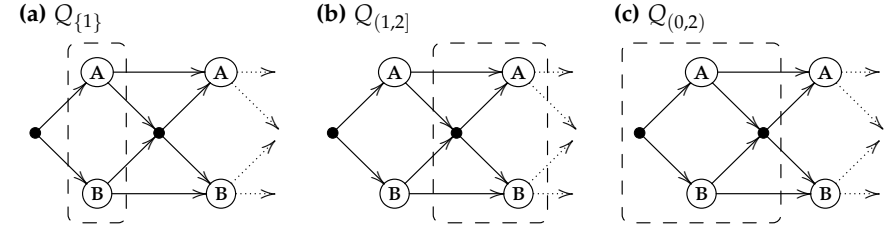
Note that for each q, q' only one line of the transition function applies. The top line, where the time index is increased, applies if q' is productive in $\mathfrak{H}$, while the bottom applies if it silent in $\mathfrak{H}$.

Unfolding has has following three properties. First, it preserves the marginal: $P_{\mathfrak{H}}(o^t) = P_{\mathcal{U}(\mathfrak{H})}(o^t)$. Second, unfolding is an idempotent operation: $\mathcal{U}(\mathfrak{H})$ is isomorphic to $\mathcal{U}(\mathcal{U}(\mathfrak{H}))$. Third, unfolding renders the set of states infinite, but for each t it preserves the number of states reachable in exactly t steps.

Order The states in an unfolded HMM have earlier-later structure. Fix $q, q' \in Q^{\mathrm{u}}$. We write $q < q'$ if there is a run to q' through q. Obviously $<$ is a partial order, furthermore it is the transitive closure of the reverse direct successor relation. It is well-founded, allowing us to perform induction on states, an essential ingredient of the forward algorithm (Algorithm 3.1) and its correctness proof (Theorem 3.3.5).

Interval Notation We introduce interval notation to address subsets of states of unfolded HMMs, as illustrated by Figure 3.5. Our notation associates each productive state with the sample size at which it produces its outcome, while the silent states fall in between. We use intervals with borders in $\mathbb{N}$. The interval contains the border $i \in \mathbb{N}$ if the addressed set of states includes the states where the i^{th} observation

Figure 3.5 Interval notation



is produced.

$$Q^{\mathrm{u}}_{[s,t)} := Q^{\mathrm{u}} \cap (Q \times [s,t)) \qquad Q^{\mathrm{u}}_{[s,t]} := Q^{\mathrm{u}}_{[s,t)} \cup Q^{\mathrm{u}}_{\{t\}}$$
$$Q^{\mathrm{u}}_{\{s\}} := Q^{\mathrm{u}} \cap (Q_{\mathrm{p}} \times \{s\}) \qquad Q^{\mathrm{u}}_{(s,t)} := Q^{\mathrm{u}}_{[s,t)} \setminus Q^{\mathrm{u}}_{\{s\}}$$
$$Q^{\mathrm{u}}_{(s,t]} := Q^{\mathrm{u}}_{[s,t]} \setminus Q^{\mathrm{u}}_{\{s\}}$$

Fix $t > 0$, then $Q^{\mathrm{u}}_{\{t\}}$ is a non-empty $<$-anti-chain (i.e. its states are pairwise $<$-incomparable). Furthermore $Q^{\mathrm{u}}_{(t,t+1)}$ is empty if $Q^{\mathrm{u}}_{\{t+1\}} = \bigcup_{q \in Q^{\mathrm{u}}_{\{t\}}} S_q$, in other words, if there are no silent states between sample sizes t and $t+1$.

3.3.4.3 The Forward Algorithm

The forward algorithm is given in Algorithm 3.1.

Analysis In Algorithm 3.1, the w array represents the weight vector at the current time. Consider a state $q \in Q$, say $q \in Q_{[t,t+1)}$. Initially, $q \notin \mathrm{dom}(w)$. Then at some point $w(q)$ is assigned the value $\mathrm{P}_\circ(q)$. This happens either in the second line because q is an initial state, or in Forward Propagation because $q \in S_{q'}$ for some predecessor q' (in this case $\mathrm{P}_\circ(q) = 0$). Then $w(q)$ accumulates weight as its direct predecessors are processed in Forward Propagation. At some point all its predecessors have been processed. If q is productive we call its weight at this point (that is, just before Loss Update) $\mathsf{alg}(\mathfrak{H}, x^{t-1}, q)$. Finally, Forward Propagation removes q from the domain of w, never to be considered again. We call the weight of q (silent or productive) just before removal $\mathsf{alg}(\mathfrak{H}, x^t, q)$.

Algorithm 3.1 Forward algorithm. Fix an unfolded deterministic HMM prior $\mathfrak{H} = \langle Q, Q_{\mathrm{p}}, \mathrm{P}_{\circ}, \mathrm{P}_{\to}, \Lambda \rangle$ on Ξ, and an $\mathcal{X}$-PFS $P_{\tilde{\xi}}$ for each expert $\xi \in \Xi$. The input consists of an infinite sequence $x_1, x_2, \ldots$ that arrives sequentially.

Declare the weight map (partial function) $w : Q \to [0,1]$.
$w(q) \leftarrow \mathrm{P}_{\circ}(q)$ **for all** q s.t. $\mathrm{P}_{\circ}(q) > 0$. ▷ $\mathrm{dom}(w) = I$
for $t = 1, 2, \ldots$ **do**
  FORWARD PROPAGATION(t)
  Predict next expert:

$$P(\boldsymbol{\xi}_t = \xi | x^{t-1}) = \sum_{q \in Q_{\{t\}} : \Lambda(q) = \xi} w(q) \Big/ \sum_{q \in Q_{\{t\}}} w(q) \,.$$

  Predict next outcome:

$$P(\boldsymbol{x}_t | x^{t-1}) = \sum_{\xi \in \Xi} P_{\tilde{\xi}}(\boldsymbol{x}_t | x^{t-1}) P(\boldsymbol{\xi}_t = \xi | x^{t-1}).$$

  LOSS UPDATE(t)
  Report probability of data: $P(x^t) = \sum_{q \in Q_{\{t\}}} w(q)$.
end for

FORWARD PROPAGATION(t)
while $\mathrm{dom}(w) \neq Q_{\{t\}}$ **do** ▷ $\mathrm{dom}(w) \subseteq Q_{[t-1,t]}$
  Pick a $<$-minimal state q in $\mathrm{dom}(w) \setminus Q_{\{t\}}$. ▷ $q \in Q_{[t-1,t)}$
  for all $q' \in S_q$ **do** ▷ $q' \in Q_{(t-1,t]}$
    $w(q') \leftarrow 0$ **if** $q' \notin \mathrm{dom}(w)$.
    $w(q') \leftarrow w(q') + w(q)\, \mathrm{P}(q \to q')$.
  end for
  Remove q from the domain of w.
end while ▷ $\mathrm{dom}(w) = Q_{\{t\}}$

LOSS UPDATE(t)
for all $q \in Q_{\{t\}}$ **do** ▷ $q \in Q_{\mathrm{p}}$
  $w(q) \leftarrow w(q) P_{\Lambda(q)}(x_t | x^{t-1})$.
end for

Note that we associate *two* weights with each productive state $q \in Q_{\{t\}}$: the weight $\mathsf{alg}(\mathfrak{H}, x^{t-1}, q)$ is calculated *before* outcome t is observed, while $\mathsf{alg}(\mathfrak{H}, x^t, q)$ denotes the weight *after* the loss update incorporates outcome t. We are now able to prove correctness of the forward algorithm.

3.3.5. Theorem. *Fix an HMM prior $\mathfrak{H}$, $t \in \mathbb{N}$ and $q \in Q_{[t,t+1]}$, then* $\mathsf{alg}(\mathfrak{H}, x^t, q) = P_{\mathfrak{H}}(x^t, q)$.

Note that the theorem applies twice to productive states: before and after production of their outcome.

Proof. By $<$-induction on states. Let $q \in Q_{(t,t+1]}$, and suppose that the theorem holds for all $q' < q$. Let $B_q = \{q' \mid \mathrm{P}(q' \to q) > 0\}$ be the set of direct predecessors of q. Observe that $B_q \subseteq Q_{[t,t+1)}$. The weight that is accumulated by Forward Propagation(t) onto q is:

$$\begin{aligned}\mathsf{alg}(\mathfrak{H}, x^t, q) &= \mathrm{P}_\circ(q) + \sum_{q' \in B_q} \mathrm{P}(q' \to q)\mathsf{alg}(\mathfrak{H}, x^t, q') \\ &= \mathrm{P}_\circ(q) + \sum_{q' \in B_q} \mathrm{P}(q' \to q) P_{\mathfrak{H}}(x^t, q') \;=\; P_{\mathfrak{H}}(x^t, q).\end{aligned}$$

The second equality follows from the induction hypothesis. Additionally if $q \in Q_{\{t\}}$ is productive, say $\Lambda(q) = \xi$, then after Loss Update(t) its weight is:

$$\begin{aligned}* \mathsf{alg}(\mathfrak{H}, x^t, q) \;&=\; P_{\tilde{\xi}}(x_t | x^{t-1}) \mathsf{alg}(\mathfrak{H}, x^{t-1}, q) \;= \\ &\qquad P_{\tilde{\xi}}(x_t | x^{t-1}) P_{\mathfrak{H}}(x^{t-1}, q) \;=\; P_{\mathfrak{H}}(x^t, q). \quad (3.9)\end{aligned}$$

The second inequality holds by induction on t, and the third by Definition 3.3.2. □

Since the algorithm computes all joint x^t, q probabilities correctly, it also correctly predicts the next outcome.

Complexity In order to analyse the time and space complexities of Algorithm 3.1, fix an HMM $\mathfrak{H}$ and $t \in \mathbb{N}$. The algorithm processes each state in $Q_{[0,t)}$ once, and at that point this state's weight is distributed over its successors. Thus, the running time is proportional to $\sum_{q \in Q_{[0,t)}} |S_q|$. The forward algorithm keeps $|\mathrm{dom}(w)|$ many weights. But

at each sample size t, $\mathrm{dom}(w) \subseteq Q_{[t,t+1]}$. Therefore the largest amount of space needed is proportional to $\max_{t'<t} |Q_{[t',t'+1]}|$.

In practice, we will mostly be interested in expressing the complexities of the algorithm in terms of two variables: the number of outcomes t, and the number of experts $k = |\Xi|$. To do so, it is convenient to extend the usual big-O notation to the bivariate case as follows: we write

$$f(t,k) = O\big(g(t,k)\big) \quad \text{if} \quad \exists c,d : \forall t,k \text{ with } t \cdot k \geq d : |f(t,k)| \leq c|g(t,k)|.$$

Turning back to our earlier examples, we find that for both Bayes (Example 3.3.3) and elementwise mixtures (Example 3.3.4) one may read from the figures that for each time t both $\sum_{q \in Q_{[t',t'+1)}} |S_q|$ and $|Q_{[t',t'+1)}|$ are $O(k)$, so both models run in $O(k\,t)$ time and require $O(k)$ space.

3.4 Regret Bounds

Here we provide some handles for analysing the predictive performance of HMMs. In each case, the idea is to compare the loss incurred by some model P to the loss incurred by another prediction strategy from a set $\mathcal{M}$ of reference strategies. For example, $\mathcal{M}$ might be the set $\{P_{\xi^t} \mid \xi^t \in \Xi^t\}$ of all prediction strategies based on a fixed expert sequence. Note that the model P is not necessarily present in $\mathcal{M}$. The goal is now to provide an upper bound on the loss overhead of the model P compared to these reference strategies. However, since we generally consider rather wide classes of reference strategies that vary significantly in complexity, it is not always possible to give a good *uniform* regret bound. Instead, the regret bound will often be expressed in terms of parameters that quantify the complexity of the reference strategy. For example, the complexity of an expert sequence ξ^t will usually be expressed in terms of the parameter m, defined as the number of contiguous blocks in ξ^t where the same expert is used.

Throughout this chapter, we use three types of regret bounds, which are given in order of increasing sophistication. The first bound is appropriate if only a few expert sequences contribute significantly to the probability of the data. In that case it is sufficient to simply drop some terms from the Bayesian mixture (3.2).

3.4.1. Lemma (Regret w.r.t. Expert Sequence ξ^t). *Let π denote an ES-prior, and let ξ^t denote a particular reference expert sequence. Then, for all data x^t,*

$$\ln \frac{P_{\xi^t}(x^t)}{P_\pi(x^t)} \;=\; \ln \frac{P_{\xi^t}(x^t)}{\sum_{\xi'^t} P_{\xi'^t}(x^t)\pi(\xi'^t)} \;\leq\; \ln \frac{P_{\xi^t}(x^t)}{P_{\xi^t}(x^t)\pi(\xi^t)} \;=\; -\ln \pi(\xi^t). \tag{3.10}$$

We obtain an expression for the regret w.r.t. some reference set $\subseteq \Xi^t$ by maximising ξ^t over its elements. We have already seen an example application: the regret bound (3.6) for $\textsc{bayes}[\Xi, w]$ is derived in this way.

In the second kind of bound, another ES-prior ρ plays the role of reference prediction strategy. It can be useful even if the number of different expert sequences with significant contribution to the probability is very large. The following lemma forms the basis for such bounds.

3.4.2. Lemma (Regret w.r.t. ES-prior ρ). *Given data x^t and ES-priors π and ρ, such that $P_\rho(x^t) > 0$. We have*

$$\ln \frac{P_\rho(x^t)}{P_\pi(x^t)} \;\leq\; -\ln \mathrm{E}_V\left[\frac{\pi(\boldsymbol{\xi}^t)}{\rho(\boldsymbol{\xi}^t)}\right] \;\leq\; \mathrm{E}_V\left[\ln \frac{\rho(\boldsymbol{\xi}^t)}{\pi(\boldsymbol{\xi}^t)}\right], \quad \textit{where} \quad V = P_\rho(\boldsymbol{\xi}^t | x^t).$$

Proof. We rewrite

$$\frac{P_\pi(x^t)}{P_\rho(x^t)} \;\geq \sum_{\xi^t : P_\rho(x^t,\xi^t)>0} \frac{P_\rho(x^t,\xi^t)}{P_\rho(x^t)} \cdot \frac{P_\pi(x^t,\xi^t)}{P_\rho(x^t,\xi^t)} \;=$$

$$\mathrm{E}_V\left[\frac{P_\pi(x^t,\boldsymbol{\xi}^t)}{P_\rho(x^t,\boldsymbol{\xi}^t)}\right] \;=\; \mathrm{E}_V\left[\frac{\pi(\boldsymbol{\xi}^t)}{\rho(\boldsymbol{\xi}^t)}\right],$$

take the $-\ln$ and subsequently apply Jensen's inequality. □

Although this bound still involves the actual data through the distribution V, sometimes the expectation can be replaced by a minimum over ξ^t. This may be sufficiently sharp for the job at hand if a good uniform bound is available for the ES-priors. However, it is possible to say more about V if the HMMs that define π and ρ share a certain structure. The next lemma is a generalisation of Theorem 1 in [129]; it is useful for *parameterised* HMMs. The HMM has to be parameterised in a specific way. Namely, for an HMM $\mathfrak{H}$, define the transition probabilities from a subset $Q^\dagger \subseteq Q$ of the state space as follows. Let $T_\eta(j) = \mathrm{e}^{\eta^\top \phi(j)} h(j)/Z(\eta)$ be an exponential family of distributions with parameter vector η, some sufficient statistic ϕ, carrier h

and normalisation $Z(\eta) = \sum_j \mathrm{e}^{\eta^\top \phi(j)} h(j)$, where j takes values in a finite set $\mathcal{J}$. Define a successor function $S : Q^\dagger \times \mathcal{J} \to Q$. Now set $\mathrm{P}(q \to q') = T_\eta(j)$, where j satisfies $S(q, j) = q'$. We also introduce the following notation. Let λ be high enough so that all runs of length λ have produced t outcomes. For each state sequence $q^\lambda \in Q^\lambda$, let $n_j(q^\lambda) = |\{i \mid 1 \leq i < \lambda, q_i \in Q^\dagger, S(q_i, j) = q_{i+1}\}|$ denote the number of transitions in q^λ between states in $Q^\dagger$ and their j-successors, and let $n(q^\lambda) = \sum_j n_j(q^\lambda)$. We can now state our result:

3.4.3. Lemma (Regret w.r.t. ML Parameter $\hat{\eta}$). *Let $\mathfrak{H}$ be parameterised as defined above and let P_η denote the joint distribution on state sequences and outcome sequences with transition probabilities from $Q^\dagger$ defined by a successor function S and an exponential family T_η, as described above. Fix outcomes x^t and let $\hat{\eta} = \operatorname{argmax}_\eta P_\eta(x^t)$. Furthermore let $W = P_{\hat{\eta}}(\boldsymbol{q}^\lambda | x^t)$ denote the posterior of $P_{\hat{\eta}}$ on runs. We then have*

$$\ln \frac{P_{\hat{\eta}}(x^t)}{P_\eta(x^t)} \;\leq\; -\ln \mathrm{E}_W \left[\frac{\pi_\eta(\boldsymbol{q}^\lambda)}{\pi_{\hat{\eta}}(\boldsymbol{q}^\lambda)} \right] \;\leq\; \mathrm{E}_W \left[\ln \frac{\pi_{\hat{\eta}}(\boldsymbol{q}^\lambda)}{\pi_\eta(\boldsymbol{q}^\lambda)} \right] \;=\; \mathrm{E}_W[n(\boldsymbol{q}^\lambda)]\, \mathrm{D}\left(T_{\hat{\eta}} \middle\| T_\eta\right),$$

where $\mathrm{D}(P\|Q) = \sum_x P(x) \log P(x)/Q(x)$ is the Kullback-Leibler divergence from P to Q.

As before, the goal of this lemma is to say something about the overhead incurred by using a particular strategy P_η instead of the reference strategy $P_{\hat{\eta}}$ The result still depends on the data via the distribution W, but in applications of the lemma the idea will be to replace $\mathrm{E}_W[n(\boldsymbol{q}^\lambda)]$ by a bound on the number of states in $Q^\dagger$ that may be traversed in any run through the HMM.

Proof of Lemma 3.4.3. The first two inequalities are Lemma 3.4.2 on the level of runs. The contribution of this lemma lies in the last equality.

First expand

$$\begin{aligned}
\mathrm{E}_W\left[\ln\frac{\pi_{\hat\eta}(\boldsymbol{q}^\lambda)}{\pi_\eta(\boldsymbol{q}^\lambda)}\right] &= \mathrm{E}_W\left[\sum_{j\in\mathcal{J}} n_j(\boldsymbol{q}^\lambda)\ln\frac{T_{\hat\eta}(j)}{T_\eta(j)}\right] \\
&= \sum_j \mathrm{E}_W[n_j(\boldsymbol{q}^\lambda)]\left((\hat\eta-\eta)^\top\phi(j)+\ln\frac{Z(\eta)}{Z(\hat\eta)}\right) \\
&= (\hat\eta-\eta)^\top\sum_j\phi(j)\,\mathrm{E}_W[n_j(\boldsymbol{q}^\lambda)] + \mathrm{E}_W[n(\boldsymbol{q}^\lambda)]\ln\frac{Z(\eta)}{Z(\hat\eta)}.
\end{aligned} \tag{3.11}$$

Since $P_{\hat\eta}$ maximises the probability $P_\eta(x^t)$ over η and since

$$\nabla_\eta \ln P_\eta(x^t, q^\lambda) = \nabla_\eta \ln\bigl(\pi_\eta(q^\lambda)/\pi_{\hat\eta}(q^\lambda)\bigr)$$

we obtain[2]

$$\vec{0} = -\nabla_\eta\frac{P_\eta(x^t)}{P_{\hat\eta}(x^t)}\Big|_{\eta=\hat\eta} = -\sum_{q^\lambda}\frac{P_{\hat\eta}(x^t,q^\lambda)}{P_{\hat\eta}(x^t)}\nabla_\eta\ln P_\eta(x^t,q^\lambda)\big|_{\eta=\hat\eta} = \nabla_\eta\,\mathrm{E}_W\left[\ln\frac{\pi_{\hat\eta}(\boldsymbol{q}^\lambda)}{\pi_\eta(\boldsymbol{q}^\lambda)}\right]\Big|_{\eta=\hat\eta}.$$

This shows that the vector differential of (3.11) must be zero at $\hat\eta$. Reordering terms we obtain

$$\sum_j\phi(j)\,\mathrm{E}_W[n_j(\boldsymbol{q}^\lambda)] = \mathrm{E}_W[n(\boldsymbol{q}^\lambda)]\nabla_\eta\ln\frac{Z(\eta)}{Z(\hat\eta)}\Big|_{\eta=\hat\eta} = \mathrm{E}_W[n(\boldsymbol{q}^\lambda)]\mathop{\mathrm{E}}_{\boldsymbol{j}\sim T_{\hat\eta}}[\phi(\boldsymbol{j})], \tag{3.12}$$

where the last step follows from

$$\nabla_\eta\ln Z(\eta) = \frac{\nabla_\eta Z(\eta)}{Z(\eta)} = \sum_{j\in\mathcal{J}}\frac{\mathrm{e}^{\eta^\top\phi(j)}h(j)}{Z(\eta)}\phi(j) = \mathop{\mathrm{E}}_{\boldsymbol{j}\sim T_\eta}[\phi(\boldsymbol{j})].$$

[2] Recall that for a function $f:\mathbb{R}^k\to\mathbb{R}$, the vector differential $\nabla_\eta f(\eta)$ is defined as the column vector $(\frac{\partial f(\eta_1)}{\partial\eta_1},\ldots,\frac{\partial f(\eta_k)}{\partial\eta_k})$.

Using (3.12) we may now simplify (3.11) to

$$\mathrm{E}_W\left[\ln \frac{\pi_{\hat{\eta}}(\boldsymbol{q}^\lambda)}{\pi_\eta(\boldsymbol{q}^\lambda)}\right] = \mathrm{E}_W[n(\boldsymbol{q}^\lambda)]\left((\eta-\hat{\eta})^\top \mathop{\mathrm{E}}_{j\sim T_{\hat{\eta}}}[\boldsymbol{\phi}(\boldsymbol{j})] + \ln \frac{Z(\eta)}{Z(\hat{\eta})}\right) = \mathrm{E}_W[n(\boldsymbol{q}^\lambda)]\,\mathrm{D}\left(T_{\hat{\eta}} \middle\| T_\eta\right),$$

completing the proof. □

As an important special case, the distribution on $\mathcal{J}$ may be fully specified by a multinomial distribution with parameter vector w; we can then apply the lemma above by setting $\phi(j) = (0,\ldots,1,0,\ldots)$, with the 1 appearing at the j^{th} position, and $h(j) = 1$. However, it may be advantageous to restrict the class of distributions on $\mathcal{J}$ to an exponential family of lower dimension, because then the reference strategy $T_{\hat{\eta}}$ has fewer degrees of freedom and the divergence $\mathrm{D}(T_{\hat{\eta}}\|T_\eta)$ that appears in the bound is decreased. We now apply the lemma to our two running examples. Note that in both cases, the model is parameterised such that the total number of parameterised transitions is known.

3.4.4. Example (Regret of Bayesian Mixtures). We have already proved the bound (3.6) for $\textsc{bayes}[\Xi, w]$ using Lemma 3.4.1, but it is instructive to do the same using Lemma 3.4.3. Let $Q^\dagger$ contain just the initial silent state and identify the experts Ξ with $\mathcal{J}$. We now have $\mathrm{E}_W[n(\boldsymbol{q}^\lambda)] = 1$, so the lemma tells us that our regret is $\mathrm{D}(\hat{w}\|w)$, where $\hat{w}$ is the *hindsight optimal* prior weight vector that maximises the probability of the available data, and w is the prior we actually use. Now observe that in order to maximise probability, $\hat{w}$ must assign all mass to a single expert $\hat{\xi}$, so $\mathrm{D}(\hat{w}\|w) = -\ln w(\hat{\xi})$ as before.

◇

3.4.5. Example (Regret of Elementwise Mixtures). We now compute the regret of $\textsc{em}[\Xi, w]$. Let $Q^\dagger$ contain the silent states and again identify the experts Ξ with $\mathcal{J}$. For elementwise mixtures, $\mathrm{E}_W[n(\boldsymbol{q}^\lambda)] = t$. So by Lemma 3.4.3, the regret of predicting the outcomes x^t with an elementwise mixture with weights w instead of the *hindsight optimal* mixture weights $\hat{w}$ is bounded by $t\,\mathrm{D}(\hat{w}\|w)$. ◇

3.5 Switching Strategies

3.5.1 Fixed Share

The paper by Herbster and Warmuth [79, 80] on *tracking the best expert* is the first to consider the scenario where the best predicting expert may change with the sample size. They by partitioning the data of size t into m segments, where each segment is associated with an expert, and give algorithms to predict almost as well as the best *partition* where the best expert is selected per segment. They give two algorithms called fixed share and variable share. The second algorithm requires a generalisation of our framework; furthermore its motivation applies only to loss functions other than log-loss. We focus on fixed share, which is in fact virtually identical to the HMM $\textsc{fs}[\Xi, w, \alpha]$ defined in Figure 3.6. Note that all arcs *into* the silent states have fixed probability $\alpha \in [0, 1]$ and all arcs *from* the silent states have some fixed distribution w on Ξ. The original algorithm uses a uniform $w(\xi) = 1/k$, and it does not allow switching to the same expert. This difference is discussed in the next section. The same algorithm is also described as an instance of the Aggregating Algorithm in [183]. Fixed share reduces to fixed elementwise mixtures by setting $\alpha = 1$ and to Bayesian mixtures by setting $\alpha = 0$. Each productive state represents that a particular expert is used at a certain sample size. Once a transition to a silent state is made, all expert history is forgotten and a new expert is chosen according to w.

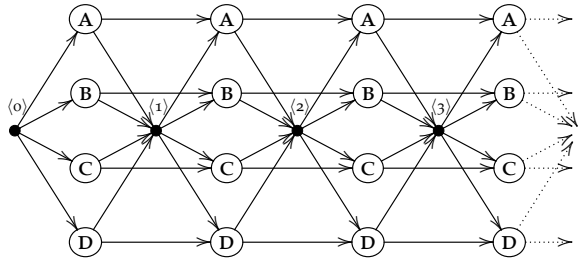
We now bound the regret of fixed share with respect to a given partition, i.e. sequence of experts.

3.5.1. Theorem (Fixed Share Regret). *Fix experts Ξ and data x^t, and let ξ^t be a sequence of experts with m blocks, $k = |\Xi|$, and $w(\xi) = 1/k$. Let $\alpha^* = (m-1)/(t-1)$ denote the switching frequency in ξ^t. Write $\mathrm{H}(\alpha^*, \alpha) = -\alpha^* \ln \alpha - (1-\alpha^*) \ln(1-\alpha)$ for the cross entropy. Then*

$$\ln \frac{P_{\xi^t}(x^t)}{P_{\textsc{fs}[\Xi,w,\alpha]}(x^t)} \;\leq\; m \ln k + (t-1)\,\mathrm{H}(\alpha^*, \alpha). \tag{3.13}$$

Proof. Let q^λ be the run that produces ξ^t and that passes through silent

Figure 3.6 Fixed share: $\textsc{fs}[\Xi, w, \alpha]$



$$Q = Q_{\mathrm{s}} \cup Q_{\mathrm{p}} \quad Q_{\mathrm{s}} = \mathbb{N} \quad Q_{\mathrm{p}} = \Xi \times \mathbb{Z}_+$$
$$\mathrm{P}_\circ(0) = 1 \quad \Lambda(\xi, t) = \xi$$
$$\mathrm{P}\begin{pmatrix} \langle t \rangle \to \langle \xi, t+1 \rangle \\ \langle \xi, t \rangle \to \langle t \rangle \\ \langle \xi, t \rangle \to \langle \xi, t+1 \rangle \end{pmatrix} = \begin{pmatrix} w(\xi) \\ \alpha \\ 1-\alpha \end{pmatrix}$$

state $\langle i \rangle$ iff $\xi_i \neq \xi_{i+1}$. Then

$$\frac{P_{\textsc{fs}[\Xi,w,\alpha]}(x^t)}{P_{\xi^t}(x^t)} \overset{\text{by (3.10)}}{\geq} \pi_{\textsc{fs}[\Xi,w,\alpha]}(\xi^t) \;\geq\; \pi_{\textsc{fs}[\Xi,w,\alpha]}(q^\lambda) \;=\; k^{-m}(1-\alpha)^{t-m}\alpha^{m-1}. \quad \square$$

Note that Herbster and Warmuth define fixed share without reflexive switches (i.e. switches to the same expert), and thus derive a bound with $\ln k + (m-1)\ln(k-1)$ instead of our $m \ln k$. We include reflexive switches in all our models to keep the exposition clean and simple, and address omitting them in Section 3.6.3.

While α^* optimises the bound, it does not necessarily maximise the probability of the data. We may wonder how much the predictive performance of the algorithm may be harmed by using α rather than the maximum likelihood value $\hat{\alpha} = \operatorname{argmax}_\alpha P_{\textsc{fs}[\Xi,w,\alpha]}(x^t)$. To this end, we can apply Lemma 3.4.3, setting $Q^\dagger$ to Q_{p}, the set of all productive states, whose outgoing transitions are parameterised by the switching rate α, to find

$$\ln \frac{P_{\textsc{fs}[\Xi,w,\hat{\alpha}]}(x^t)}{P_{\textsc{fs}[\Xi,w,\alpha]}(x^t)} \;\leq\; (t-1)\,\mathrm{D}\left(\hat{\alpha} \| \alpha\right). \tag{3.14}$$

Judging from (3.13) and (3.14), the regret appears to grow linearly with time, but if we substitute the switching rate $\alpha = \alpha^*$ that optimises the bound, cross entropy reduces to ordinary entropy, and we find that the regret only has a logarithmic dependence on t: we have

$$(m-1)\ln\frac{t-1}{m-1} \quad \leq \quad (t-1)H(\alpha^*) \quad \leq \quad (m-1)\ln\frac{t-1}{m-1} + m. \tag{3.15}$$

The problem is that such asymptotics can only be achieved if we are somehow able to guess the optimal switching rate before observing the data. This issue is addressed in the following sections. We will evaluate the performance of the other models for expert tracking using the loss of fixed share with $\alpha = \alpha^*$ as a baseline.

3.5.2 Intermezzo: Interpolation

Note how Fixed share (Figure 3.6) interpolates between the Bayesian Mixture (Figure 3.2) and the Elementwise Mixture (Figure 3.3).The parameter α determines *when* switches occur. If no switch occurs then the Bayesian Mixture's transitions are used: all experts' weights remain unchanged. On the other hand, if a switch occurs then the Elementwise Mixture's transitions are used: all experts' weights are gathered and redistributed.

Interpolations are natural to the switching domain. In [41], so-called Bernoulli HMMs are used to produce switching rates, whereas in Chapter 4 a fixed switching rate α is used, varying instead the HMMs that specify the normal and switching behaviour.

We now describe the general interpolation mechanism, which takes three HMMs, $\mathfrak{H}$, $\mathbb{B}_{\mathsf{n}}$ and $\mathbb{B}_{\mathsf{s}}$. The *interpolator* $\mathfrak{H}$ specifies *when* switches occur, $\mathbb{B}_{\mathsf{n}}$ determines the *normal* (n) evolution and $\mathbb{B}_{\mathsf{s}}$ determines the evolution when a *switch* (s) occurs. In particular, we obtain a model that defines the same distribution as $\textsc{fs}[\Xi, w, \alpha]$ by interpolation using $\mathfrak{H} = \textsc{em}[\{\mathsf{n},\mathsf{s}\}, (1-\alpha, \alpha)]$, $\mathbb{B}_{\mathsf{n}} = \textsc{bayes}[\Xi, w]$ and $\mathbb{B}_{\mathsf{s}} = \textsc{em}[\Xi, w]$.

3.5.2. Definition (Interpolation). See Figure 3.7 for an illustration. Let $\mathfrak{H} = \langle Q^{\mathfrak{H}}, Q_{\mathrm{p}}^{\mathfrak{H}}, \mathrm{P}_{\circ}^{\mathfrak{H}}, \mathrm{P}_{\rightarrow}^{\mathfrak{H}}, \Lambda^{\mathfrak{H}}\rangle$ be a deterministic unfolded HMM on $\{\mathsf{n},\mathsf{s}\}$, and let $\mathbb{B}_{\mathsf{n}}$ and $\mathbb{B}_{\mathsf{s}}$ be deterministic unfolded HMMs on experts Ξ sharing a common state set Q with identical initial distribution $\mathrm{P}_{\circ}$ and interpretation Λ (and thus identical productive states Q_{p}). We define

$\mathbb{B}_\mathsf{n} \otimes_\mathfrak{H} \mathbb{B}_\mathsf{s}$, the $\mathfrak{H}$-*interpolation of* $\mathbb{B}_\mathsf{n}$ *and* $\mathbb{B}_\mathsf{s}$, by

$$\mathbb{B}_\mathsf{n} \otimes_\mathfrak{H} \mathbb{B}_\mathsf{s} \;\coloneqq\; \big\langle Q^\otimes, Q^\otimes_\mathrm{p}, \mathrm{P}^\otimes_\circ, \mathrm{P}^\otimes_\to, \Lambda^\otimes \big\rangle.$$

Each state of the interpolation is a pair of states, consisting of one state from either $\mathbb{B}$ HMM, and one state from the interpolator $\mathfrak{H}$, at least one of them productive:

$$Q^\otimes \;\coloneqq\; Q \times Q^\mathfrak{H}_\mathrm{p} \cup Q_\mathrm{p} \times Q^\mathfrak{H},$$

and productive states of the interpolation are the pairs with two contemporary productive states

$$Q^\otimes_\mathrm{p} \;\coloneqq\; \bigcup_{t \geq 1} Q_{\{t\}} \times Q^\mathfrak{H}_{\{t\}}.$$

In the following we assume w.l.o.g. that $\mathrm{P}_\circ$ and $\mathrm{P}^\mathfrak{H}_\circ$ are on productive states. The initial distribution $\mathrm{P}^\otimes_\circ$ independently chooses a productive state q from Q using $\mathrm{P}_\circ$, and a productive state a from $Q^\mathfrak{H}$ using the initial distribution $\mathrm{P}^\mathfrak{H}_\circ$

$$\mathrm{P}^\otimes_\circ\big(\langle q, a \rangle\big) \;\coloneqq\; \mathrm{P}_\circ(q)\, \mathrm{P}^\mathfrak{H}_\circ(a).$$

The transition function $\mathrm{P}^\otimes_\to$ first forwards the first state component (q in Q) to the next productive state in Q using either $\mathrm{P}^{\mathbb{B}_\mathsf{n}}_\to$ or $\mathrm{P}^{\mathbb{B}_\mathsf{s}}_\to$ as determined by the produced label $\Lambda^\mathfrak{H}(a) \in \{\mathsf{n}, \mathsf{s}\}$. Then it forwards the second state component (a in $Q^\mathfrak{H}$) to the next productive state using $\mathrm{P}^\mathfrak{H}_\to$. Abbreviating $\mathbb{B}_{\Lambda^\mathfrak{H}(a)}$ to $\mathbb{B}_a$ we have

$$\mathrm{P}^\otimes_\to\big(\langle q, a \rangle \to \langle q', a' \rangle\big) \;\coloneqq\; \begin{cases} \mathrm{P}^{\mathbb{B}_a}_\to(q \to q') & \text{if } q \in Q_{[t,t+1)},\ a = a' \in Q^\mathfrak{H}_{\{t\}}, \\ \mathrm{P}^\mathfrak{H}_\to(a \to a') & \text{if } a \in Q^\mathfrak{H}_{[t,t+1)},\ q = q' \in Q_{\{t+1\}}, \\ 0 & \text{otherwise.} \end{cases}$$

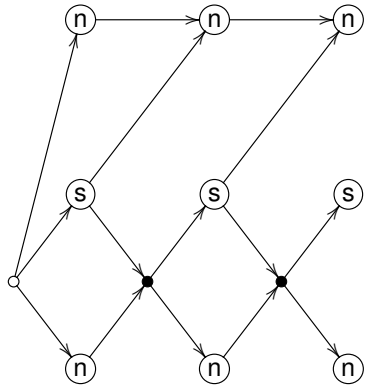
Finally, the node label is that of the first component

$$\Lambda^\otimes\big(\langle q, a \rangle\big) \;\coloneqq\; \Lambda(q).$$

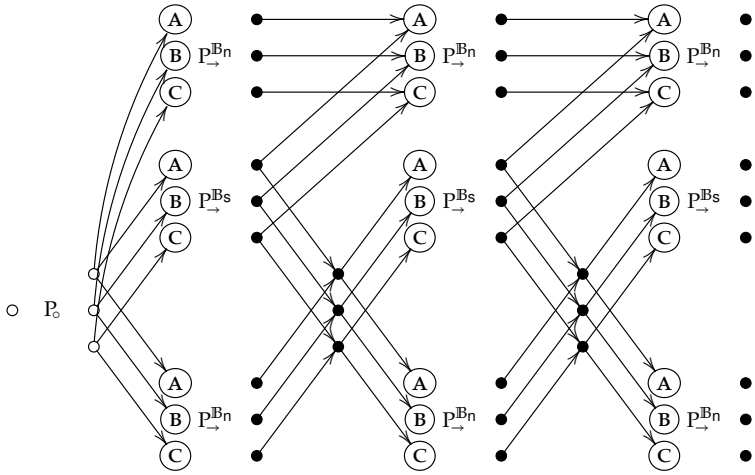
Figure 3.7b shows the state transition diagram of an interpolation, with the interpolator shown in Figure 3.7a. Figure 3.7c displays the Bayesian

Figure 3.7 Interpolation example: graph structure

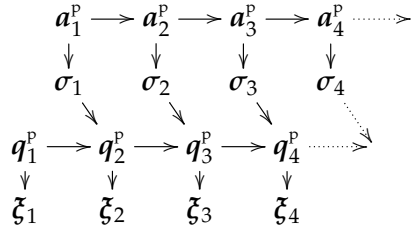
(a) HMM $\mathfrak{H}$ on $\{n, s\}$ (normal/switch)



(b) Interpolation HMM $\mathbb{B}_n \otimes_{\mathfrak{H}} \mathbb{B}_s$ on experts $\{A, B, C\}$



(c) Bayesian network of interpolation



network of an interpolation. The random variables $\boldsymbol{q}_i^{\mathrm{p}}$ and $\boldsymbol{a}_i^{\mathrm{p}}$ are the components of the productive state at time i, while $\boldsymbol{\xi}_i = \Lambda(\boldsymbol{q}_i^{\mathrm{p}})$ and $\boldsymbol{\sigma}_i = \Lambda^{\mathfrak{H}}(\boldsymbol{a}_i^{\mathrm{p}})$. Note that $\boldsymbol{\sigma}_i = \mathsf{s}$ if a switch occurs between time i and $i+1$.

Interpolation separates concerns; $\mathfrak{H}$, on the highest level, determines when to switch. Below, $\mathbb{B}_{\mathsf{n}}$ and $\mathbb{B}_{\mathsf{s}}$ determine the normal and switching behaviour. This separation is reflected in the following modular loss bound:

3.5.3. Lemma (Interpolation Decomposition). *Abbreviate $\pi_{\mathbb{B}_\sigma}$ to π_σ. For each sequence $\sigma^{t-1} \in \{\mathsf{n},\mathsf{s}\}^{t-1}$ of switch decisions (on the $\mathfrak{H}$ level) and each sequence $q_{\mathrm{p}}^t \in Q_{\mathrm{p}}^t$ of productive states (on the $\mathbb{B}$ level)*

$$\pi_{\mathbb{B}_{\mathsf{n}} \otimes_{\mathfrak{H}} \mathbb{B}_{\mathsf{s}}}(q_{\mathrm{p}}^t) \;\geq\; \pi_{\mathfrak{H}}(\sigma^{t-1})\; \mathrm{P}_\circ(q_1^{\mathrm{p}}) \prod_{i=1}^{t-1} \pi_{\sigma_i}(q_{i+1}^{\mathrm{p}} | q_i^{\mathrm{p}}).$$

Proof. For every distribution

$$\pi_{\mathbb{B}_{\mathsf{n}} \otimes_{\mathfrak{H}} \mathbb{B}_{\mathsf{s}}}(q_{\mathrm{p}}^t) \;\geq\; \pi_{\mathbb{B}_{\mathsf{n}} \otimes_{\mathfrak{H}} \mathbb{B}_{\mathsf{s}}}(\sigma^{t-1}) \pi_{\mathbb{B}_{\mathsf{n}} \otimes_{\mathfrak{H}} \mathbb{B}_{\mathsf{s}}}(q_{\mathrm{p}}^t | \sigma^{t-1}).$$

By the definition of interpolation, we have $\pi_{\mathbb{B}_{\mathsf{n}} \otimes_{\mathfrak{H}} \mathbb{B}_{\mathsf{s}}}(\sigma^{t-1}) = \pi_{\mathfrak{H}}(\sigma^{t-1})$ and $\pi_{\mathbb{B}_{\mathsf{n}} \otimes_{\mathfrak{H}} \mathbb{B}_{\mathsf{s}}}(q_{\mathrm{p}}^t | \sigma^{t-1}) = \mathrm{P}_\circ(q_1^{\mathrm{p}}) \prod_{i=1}^{t-1} \pi_{\sigma_i}(q_{i+1}^{\mathrm{p}} | q_i^{\mathrm{p}})$. □

3.5.4. Corollary (Default Interpolation Regret). *We apply this theorem to our $\mathbb{B}$-level HMMs of interest, $\mathbb{B}_{\mathsf{n}} = \textsc{bayes}[\Xi, w]$ and $\mathbb{B}_{\mathsf{s}} = \textsc{em}[\Xi, w]$ where w is uniform on k experts. Fix ξ^t. Set $\sigma_i = \mathsf{s}$ iff $\xi_{i+1} \neq \xi_i$, and let m be the number of blocks in ξ^t, i.e. the number of s in σ^{t-1} plus one. Then for all data x^t*

$$\ln \frac{P_{\xi^t}(x^t)}{P_{\mathbb{B}_{\mathsf{n}} \otimes_{\mathfrak{H}} \mathbb{B}_{\mathsf{s}}}(x^t)} \;\leq\; -\ln \pi_{\mathfrak{H}}(\sigma^{t-1}) + m \ln k.$$

Proof. Recall that in both $\mathbb{B}$-level HMMs there is, at each time, a one-one correspondence between productive states and experts, so we may just as well identify them. Then we have $\mathrm{P}_\circ(\xi_1) = w(\xi_1)$,

$$\pi_{\textsc{bayes}[\Xi,w]}(\xi_i = \xi_{i-1} | \xi_{i-1}) = 1, \text{ and} \qquad \pi_{\textsc{em}[\Xi,w]}(\xi_i | \xi_{i-1}) = w(\xi_i).$$

Lemma 3.5.3, using $w(\xi) = 1/k$, yields $\pi_{\mathbb{B}_{\mathsf{n}} \otimes_{\mathfrak{H}} \mathbb{B}_{\mathsf{s}}}(\xi^t) \geq \pi_{\mathfrak{H}}(\sigma^{t-1})\; k^{-m}$ and the result follows by (3.10). □

3.5.5. Example (Fixed Share Regret). We shorten $\textsc{em}[\{\mathsf{n},\mathsf{s}\},(1-\alpha,\alpha)]$ to $\textsc{fs}[\alpha]$. We now redefine the fixed share model in terms of the interpolation

$$\textsc{fs}[\Xi,w,\alpha] \;:=\; \textsc{bayes}[\Xi,w] \otimes_{\textsc{fs}[\alpha]} \textsc{em}[\Xi,w].$$

Note that this definition is equivalent to the one in Figure 3.6, as the sets of infinite runs (of states) are in one-one correspondence between the models, and so in particular they induce the same ES-joint. We now reprove the fixed share regret bound (3.13) by combining Corollary 3.5.4 with the observation that

$$-\ln \pi_{\textsc{fs}[\alpha]}(\sigma^{t-1}) \;=\; -\ln\big((1-\alpha)^{t-m}\alpha^{m-1}\big) \;=\; (t-1)\,\mathrm{H}(\alpha^*,\alpha). \quad (3.16)$$

In the following we often use this mechanism. We prove a loss bound for the interpolator $\mathfrak{H}$ on switch sequences, and transport it to the data level using Corollary 3.5.4, adding $m \ln k$. ◇

Running Time of Forward Algorithm The forward algorithm (Algorithm 3.1, Section 3.3.4.3) decomposes for interpolations. To compute the round t forward propagation step on the interpolation $\mathbb{B}_\mathsf{n} \otimes_\mathfrak{H} \mathbb{B}_\mathsf{s}$, we need to compute $|Q_{\{t\}}|$ many forward propagations on $\mathbb{B}_\mathsf{s}/\mathbb{B}_\mathsf{n}$, followed by $|Q^\mathfrak{H}_{\{t\}}|$ many forward propagations on $\mathfrak{H}$. For our HMMs of interest, the total running time of the forward algorithm on the interpolation is dominated by $|\Xi|$ times the running time on $\mathfrak{H}$.

Outlook This concludes the intermezzo. In the remainder of this section, we discuss the benefits and costs of several choices for $\mathfrak{H}$, both in terms of loss bound and in terms of running time. We also briefly discuss alternatives for $\mathbb{B}_\mathsf{n}$ and $\mathbb{B}_\mathsf{s}$.

3.5.3 Decreasing Switching Rate

Fixed share uses a fixed switching rate α. However, it is possible to get good bounds without having to choose α, by letting the switching probability decrease as a function of time. This approach was invented in our group in CWI, Amsterdam, but from personal discussion with Mark Herbster we learned that he independently invented an algorithm similar to the slowly decreasing switching rate (Section 3.5.3.1), as early as 1997. Fixed share uses the elementwise mixture interpolator with

switching rate α. We consider a new interpolator, $\textsc{dsr}[\alpha^\omega]$, which is similar to $\textsc{fs}[\alpha]$, except that the switching probability α_t is no longer a parameter of the model, but a fixed decreasing function of the time t. We still model switches as independent, and as before, we define the full model as

$$\textsc{dsr}[\Xi, w, \alpha^\omega] \;:=\; \textsc{bayes}[\Xi, w] \otimes_{\textsc{dsr}[\alpha^\omega]} \textsc{em}[\Xi, w].$$

To obtain bounds, we use the following equality. Let σ^{t-1} be a sequence with $m-1$ occurrences of s at positions $t_2, \ldots, t_m$, and let $t_1 = 0$. Then

$$\begin{aligned} -\ln \pi_{\textsc{dsr}[\alpha^\omega]}(\sigma^{t-1}) \;&=\; -\ln\left(\prod_{i=1}^{t-1}(1-\alpha_i)\prod_{j=2}^{m}\frac{\alpha_{t_j}}{1-\alpha_{t_j}}\right) \;= \\ &\qquad -\sum_{i=1}^{t-1}\ln(1-\alpha_i) - \sum_{j=2}^{m}\ln\frac{\alpha_{t_j}}{1-\alpha_{t_j}}. \end{aligned} \tag{3.17}$$

The middle expression can be read as follows: the first sum denotes the cost of not switching during the first t outcomes, and the second sum denotes the correction for the switches that actually did occur.

We now consider two interesting choices for the switching rate α_i, solving the two problems that we identified for fixed share, namely that the α parameter has to be tuned, and that the regret keeps increasing even if, from some point on, no switches occur anymore.

3.5.3.1 Switching with Slowly Decreasing Probability

3.5.6. Theorem. *Let $\alpha_i = 1 - \mathrm{e}^{-c/i}$ for some $c > 0$. Let w be the uniform distribution on the set Ξ of k experts. For any data x^t and expert sequence ξ^t with m blocks*

$$\ln\frac{P_{\xi^t}(x^t)}{P_{\textsc{dsr}[\Xi,w,\alpha^\omega]}(x^t)} \;\le\; m\ln k + c - (m-1)\ln c + (m-1+c)\ln(t-1). \tag{3.18}$$

Proof. By (3.17), using $\sum_{i=1}^{t} \frac{1}{i} < \ln t + 1$ and $e^x \geq x + 1$,

$$-\ln \pi_{\mathrm{DSR}[\alpha^\omega]}(\sigma^{t-1}) \;=\; c \sum_{i=1}^{t-1} \frac{1}{i} - \sum_{j=2}^{m} \ln\bigl(e^{c/t_j} - 1\bigr) \;\leq$$

$$c \ln(t-1) + c - (m-1)\ln c + \sum_{j=2}^{m} \ln t_j.$$

The sum is bounded by substituting each t_j by $t-1$, and the result follows by Corollary 3.5.4. □

Note that while we succeeded in eliminating the parameter α, we have in fact introduced a new parameter c, so it would appear that matters have not improved much. But in fact, as c does not appear in the dominant term of the bound, it may safely be set to some convenient constant such as $c = 1$ or $c = 1/e$. The optimising value is $c^* = (m-1)/(1 + \ln(t-1))$, which yields slightly better asymptotics, but this defeats the purpose as it would require a priori knowledge of m and t again.

We now compare the regret bound (3.18) to the bound (3.16) for fixed share. To maximise the difference, we use the optimising parameter α^* for fixed share, and we lower bound the entropy using (3.15). The difference is

$$c - (m-1)\ln c + c \ln(t-1) + (m-1)\ln(m-1),$$

where the last two terms dictate asymptotic behaviour. Which of these terms is dominant depends on how quickly m grows as a function of t. If there are relatively few switches, $m \ln m = o(\ln t)$, then the $c \ln(t-1)$ term dominates, so it pays to use a small value for c to get good asymptotics in that case. If, on the other hand, the number of switches is large, then the last term is larger, and it may be substantial; careful judgement is then required to decide whether or not this is an acceptable price to pay or that a more sophisticated method for *learning* the switching rate (Section 3.5.4) is preferable.

3.5.3.2 Switching with More Quickly Decreasing Probability

In some settings the optimal number of switches between experts may remain bounded. For example, in [177] the considered experts are

Bayesian prediction strategies associated with parametric models of varying complexity; at small sample sizes, simple models typically make the best predictions, but if one of the more complex models contains (a distribution closest to) the data generating distribution, then one expects that model to eventually make the best predictions. From that point in time onwards, no more switches away from that model are required.

In such a scenario, a simple Bayesian combination of the experts with uniform prior yields a regret bound of $\ln k$ w.r.t. the ultimately best expert (see (3.6)), which depends on the number of experts but *not on the sample size*. Asymptotically, this is therefore a better solution than the one presented in the previous section, where even if there are no switches at all ($m = 1$), the incurred regret bound of $\ln k + c + c \ln(t - 1)$ grows without bound. This happens because the ES-prior $\textsc{dsr}[\alpha^\omega]$ assigns zero probability to the event that no more switches occur from some time t onwards.

There are two ways to tweak the model somewhat to ensure that the probability of no more switches is strictly positive. This section considers the simplest approach, which is just to let the probability of switching decrease slightly faster. A different method called the *switch distribution*, introduced in [177], is briefly discussed in the next section.

3.5.7. Theorem. *Let $\alpha_i = 1 - \mathrm{e}^{-c\tau(t)}$ for some $c > 0$ and a decreasing mass function τ on the positive integers. Let w be the uniform distribution on the set Ξ of k experts. For any data x^t and expert sequence ξ^t with m blocks*

$$\ln \frac{P_{\xi^t}(x^t)}{P_{\textsc{dsr}[\Xi,w,\alpha^\omega]}(x^t)} \;\leq\; m \ln k + c - (m-1)\ln c - (m-1)\ln \tau(t_m). \quad (3.19)$$

Proof. Using (3.17), $\sum_i \tau(i) = 1$ and $\mathrm{e}^x \geq x + 1$,

$$-\ln \pi_{\textsc{dsr}[\alpha^\omega]}(\sigma^{t-1}) \;=\; c \sum_{i=1}^{t-1} \tau(i) - \sum_{j=2}^{m} \ln\bigl(\mathrm{e}^{c\cdot\tau(t_j)} - 1\bigr) \;\leq\; c - (m-1)\ln c - \sum_{j=2}^{m} \ln \tau(t_j).$$

For decreasing τ, we obtain an upper bound by substituting $t_i = t_m$ for $1 \leq i < t_m$, and the theorem follows from Corollary 3.5.4. □

A desirable feature of this bound is that it is expressed in terms of the index t_m of the last switch rather than in terms of the time t. The role of c is even weaker than before, since it no longer features in a $c \ln t$ penalty term; its optimal value is now $c^* = m - 1$, meaning that a value of 1 or larger will generally be sensible. To choose a suitable prior τ, note that the bound depends on the prior probability of t_m, which is typically at least moderately large. Therefore it is sensible to use a fat-tailed prior. A convenient choice is

$$\tau(t) = \frac{1}{\ln(t+e-1)} - \frac{1}{\ln(t+e)}, \tag{3.20}$$

which satisfies

$$-\ln \tau(t) \leq \ln(t) + 2 \ln \ln(t+e) + e/t.$$

To compare the resulting bound to the bound from Section 3.5.3.1, assume $t_m = t - 1$ and overestimate (3.19) by

$$m \ln k + c - (m-1) \ln c + (m-1) \ln(t-1) + 2(m-1) \ln \ln(t-1+e) + e.$$

Subtracting (3.18) we get a difference of

$$2(m-1) \ln \ln(t-1+e) - c \ln(t-1) + e.$$

Thus, asymptotically the new bound (3.19) improves upon (3.18) if the number of switches m does not grow too quickly as a function of t, to be precise if $m \ln \ln t = o(\ln t)$. The current choice of α^ω has the simultaneous advantage of bounded regret w.r.t. reference expert sequences with a bounded number of switches.

3.5.3.3 The Switch Distribution

Like the model with quickly decreasing probability of switching, the *switch distribution* is a model with the feature that the regret bound is parameterised by the index of the last switch t_m rather than the time t. It is an adaptation of the model with *slowly* decreasing switching probability (Section 3.5.3.1). The structure of its defining interpolator is displayed in Figure 3.7a. The idea is that with every switch, there is a certain fixed probability of "stabilisation", meaning that the interpolator enters a special "band" of states where further switching is

impossible. The resulting loss bound is very similar to (3.18) except that t is replaced by $t_m + 1$, and there is an additional stabilisation penalty of $-(m-1)\ln(1-\theta) - \ln\theta$ where $0 < \theta < 1$ is some fixed stabilisation probability.

The switch distribution was developed for the purpose of MDL and Bayesian model selection and model averaging. In [177] the switch distribution is shown to achieve the optimal *rate of convergence* when used for sequential prediction, but at the same time, it defines a model selection criterion that can be shown to be *consistent* (select the model containing the true distribution with probability 1 as sufficient data become available).

In fact, the results in [177] apply also to the model with quickly decreasing switching probability of Section 3.5.3.2, which is significantly simpler. For further details of how the original switch distribution can be cast as an HMM, including a proof that this HMM corresponds to the parametric definition of the ES-prior, the reader is referred to [101]. An abbreviated, but more polished, discussion appears in [100].

3.5.4 Learning the Switching Rate

3.5.4.1 The Switching Method

In a very early publication, Volf and Willems [180] describe an algorithm called *the switching method*, which is very similar to Herbster and Warmuth's fixed share, except that it is able to learn the optimal switching rate α online. Here we describe it as an interpolation and bound its regret. Whereas fixed share interpolates using a fixed Bernoulli$[\alpha]$ distribution, the switching method "integrates out" the parameter using Jeffreys' prior (which is Beta$[\frac{1}{2}, \frac{1}{2}]$).

The switching method HMM is defined as the interpolation

$$\textsc{sm}[\Xi, w] \;:=\; \textsc{bayes}[\Xi, w] \otimes_{\textsc{sm}} \textsc{em}[\Xi, w],$$

with the interpolator SM defined in Figure 3.8. Each productive state $\langle n_{\mathsf{n}}, n_{\mathsf{s}}, \sigma\rangle$ represents the fact that after observation $n_{\mathsf{n}} + n_{\mathsf{s}} + 1$ a switch occurs ($\sigma = \mathsf{s}$) or not ($\sigma = \mathsf{n}$), while there have been n_{s} switches in the past.

We now bound the regret of the switching method with respect to fixed share with any switching rate α (in particular the maximum

likelihood rate $\hat{\alpha}$), and thereby show that it is universal for the fixed-share model class $\left\{ P_{\text{FS}[\Xi,w,\alpha]} \;\middle|\; \alpha \in [0,1] \right\}$. As far as we know, this bound is new.

3.5.8. Theorem (The Switching Method Regret). *For any switching rate α and data x^t*

$$\ln \frac{P_{\text{FS}[\Xi,w,\alpha]}(x^t)}{P_{\text{SM}[\Xi,w]}(x^t)} \;\leq\; \ln 2 + \tfrac{1}{2} \ln t.$$

Proof. Fixed share and the switching method interpolate the same underlying HMMs, so we have the following information processing inequalities (c.f. Lemma 3.4.2)

$$\max_{x^t} \frac{P_{\text{FS}[\Xi,w,\alpha]}(x^t)}{P_{\text{SM}[\Xi,w]}(x^t)} \;\leq\; \max_{\xi^t} \frac{P_{\text{FS}[\Xi,w,\alpha]}(\xi^t)}{P_{\text{SM}[\Xi,w]}(\xi^t)} \;\leq\; \max_{\sigma^{t-1}} \frac{P_{\text{FS}[\Xi,w,\alpha]}(\sigma^{t-1})}{P_{\text{SM}[\Xi,w]}(\sigma^{t-1})} \;=\; \max_{\sigma^{t-1}} \frac{P_{\text{FS}[\alpha]}(\sigma^{t-1})}{P_{\text{SM}}(\sigma^{t-1})}.$$

Thus we may transfer regret bounds from the interpolator level via the expert-sequence level to the data level. The rightmost term is the worst-case regret for the Bernoulli model with Jeffreys prior, which can be bounded (see e.g. [190]) by $\ln 2 + \frac{1}{2} \ln t$ for all α. □

By the above theorem and the fixed share regret bound Theorem 3.5.1, we obtain for all ξ^t with switching rate α^*

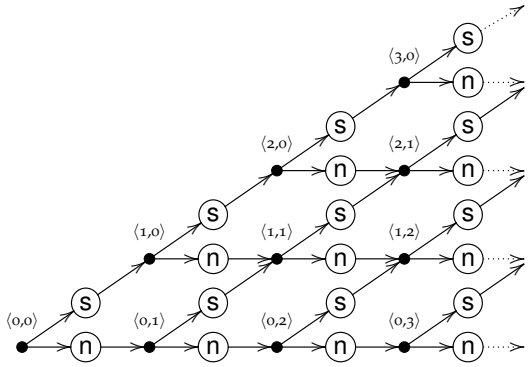
$$\ln \frac{P_{\xi^t}(x^t)}{P_{\text{SM}[\Xi,w]}(x^t)} \;\leq\; m \ln k + (t-1)\, \mathrm{H}(\alpha^*) + \ln 2 + \tfrac{1}{2} \ln t.$$

The switching method was independently derived by [18], who also proved the above bound. Our theorem is slightly sharper, as it bounds the regret w.r.t. the maximum-likelihood fixed-share performance instead of its regret bound.

3.5.4.2 Improving Time Efficiency for Learning the Switching Rate

The new ingredient of the switching method compared to fixed share is that the HMM includes a switch count in each state. This allows us to adapt the switching probability to the data, but it also renders

Figure 3.8 The switching method interpolator SM



$$Q = Q_{\mathsf{s}} \cup Q_{\mathsf{p}} \quad Q_{\mathsf{s}} = \mathbb{N}^2 \quad Q_{\mathsf{p}} = \mathbb{N}^2 \times \{\mathsf{n}, \mathsf{s}\}$$
$$\Lambda(n_{\mathsf{n}}, n_{\mathsf{s}}, \sigma) = \sigma \quad \mathrm{P}_\circ(0,0) = 1$$
$$\mathrm{P}_{\rightarrow} \begin{pmatrix} \langle n_{\mathsf{n}}, n_{\mathsf{s}}, \mathsf{n}\rangle \to \langle n_{\mathsf{n}}+1, n_{\mathsf{s}}\rangle \\ \langle n_{\mathsf{n}}, n_{\mathsf{s}}, \mathsf{s}\rangle \to \langle n_{\mathsf{n}}, n_{\mathsf{s}}+1\rangle \\ \langle n_{\mathsf{n}}, n_{\mathsf{s}}\rangle \to \langle n_{\mathsf{n}}, n_{\mathsf{s}}, \mathsf{n}\rangle \\ \langle n_{\mathsf{n}}, n_{\mathsf{s}}\rangle \to \langle n_{\mathsf{n}}, n_{\mathsf{s}}, \mathsf{s}\rangle \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ \frac{(n_{\mathsf{n}}+\frac{1}{2})}{(n_{\mathsf{n}}+n_{\mathsf{s}}+1)} \\ \frac{(n_{\mathsf{s}}+\frac{1}{2})}{(n_{\mathsf{n}}+n_{\mathsf{s}}+1)} \end{pmatrix}$$

the number of states quadratic. The quadratic running time $O(k\,t^2)$ restricts its use to moderately sized data sets. Monteleoni and Jaakkola [129] place a *discrete* prior on the switching rate α: the prior mass is distributed over $\sqrt{t}$ well-chosen points, where the ultimate sample size t is assumed known. This way they still achieve the bound of Theorem 3.5.8 up to a constant, while reducing the running time to $O(k\,t\sqrt{t})$.

The approach taken by Monteleoni and Jaakkola has two disadvantages of its own: first, the ultimate sample size t has to be known in advance, which means that the presented algorithm is only quasi-online. Second, the discretisation of the prior is defined only algorithmically, which means that both the number and the values of the discretisation points are not known symbolically. As a consequence, the resulting regret bound can only be determined up to $O(1)$. In [41] a simple *explicit* discretisation scheme is presented which allows the regret bound to be calculated exactly. Furthermore, it is shown how, at the cost of a

somewhat worse regret bound, this discretisation scheme can be *refined* online such that t no longer has to be known in advance.

3.5.5 The Run-length Model for Clustered Switching

Run-length codes have been used extensively in the context of data compression, see e.g. [128]. Rather than applying run length codes directly to the observations, we use the corresponding probability distributions on binary sequences as interpolators, as they may constitute good models for the distances between consecutive switches.

The run-length model is especially useful if the switches are clustered, in the sense that some parts of the expert sequence contain relatively few switches, while other parts contain many. The fixed share algorithm remains oblivious to such properties, as its interpolator is a Bernoulli model: the probability of switching remains the same, regardless of the index of the previous switch. Essentially the same limitation also applies to the switching method, whose switching probability normally converges as the sample size increases. The switch distribution is efficient when the switches are clustered toward the beginning of the sample: its switching probability decreases in the sample size. However, this may be unrealistic and may introduce a new unnecessary loss overhead.

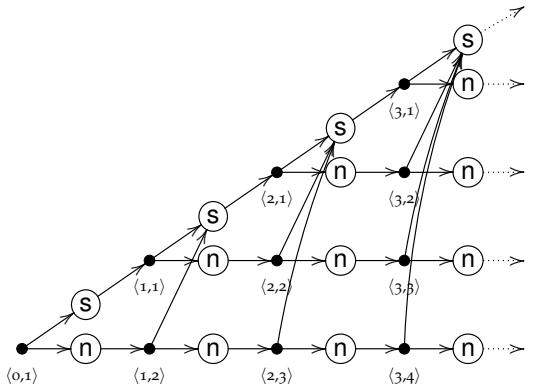
The run-length model is based on the assumption that the *intervals* between successive switches are independently distributed according to some distribution τ. After the universal share model and the switch distribution, this is a third generalisation of the fixed share algorithm, which is recovered by taking a geometric distribution for τ.

Let τ be a distribution on $\mathbb{Z}_+ \cup \{\infty\}$, which is used to model the lengths of the blocks. We assume $\tau(\infty) > 0$; this keeps our regret constant when the reference number of switches is bounded while the number of samples goes to infinity. The run-length interpolator $\textsc{rl}[\tau]$ is defined in Figure 3.9. Intuitively, the state $\langle t, \delta \rangle$ means that we are at time t, and that sample $t+1$ will be the δth sample since the last switch. The HMM for the run-length model is given by the interpolation

$$\textsc{rl}[\Xi, w, \tau] \;:=\; \textsc{bayes}[\Xi, w] \otimes_{\textsc{rl}[\tau]} \textsc{em}[\Xi, w].$$

As may be read from the diagram of the interpolator, evaluating the run-length model requires quadratic running time $O(k\, t^2)$ in general.

Figure 3.9 The run-length model interpolator $\textsc{rl}[\tau, c]$



$$Q = Q_{\mathsf{s}} \cup Q_{\mathsf{p}} \quad Q_{\mathsf{s}} = \mathbb{S} \quad Q_{\mathsf{p}} = \{\mathsf{n}\} \times \mathbb{S} \cup \{\mathsf{s}\} \times \mathbb{N}$$

$$\mathrm{P}_{\circ}(0,1) = 1 \quad \Lambda(\mathsf{n}, t, \delta) = \mathsf{n} \quad \Lambda(\mathsf{s}, t) = \mathsf{s}$$

$$\mathrm{P}_{\rightarrow} \begin{pmatrix} \langle \mathsf{s}, t\rangle \to \langle t, 1\rangle \\ \langle \mathsf{n}, t, \delta\rangle \to \langle t, \delta\rangle \\ \langle t, \delta\rangle \to \langle \mathsf{n}, t{+}1, \delta{+}1\rangle \\ \langle t, \delta\rangle \to \langle \mathsf{s}, t{+}1\rangle \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ \tau\big(z > \delta | z \geq \delta\big) \\ \tau\big(z = \delta | z \geq \delta\big) \end{pmatrix}$$

where

$$\mathbb{S} \coloneqq \left\{ \langle t, \delta\rangle \in \mathbb{N}^2 \mid \delta \leq t + 1 \right\}.$$

3.5.9. Theorem (Run-length Model Regret). *Let w be the uniform distribution on k experts. Assume there is a log-convex function ϑ on $[1, \infty)$ that agrees with τ on $\mathbb{Z}_+$. With abuse of notation, we identify τ with ϑ. Then, for all data x^t and expert sequences ξ^t with m blocks, we have*

$$\ln \frac{P_{\xi^t}(x^t)}{P_{\textsc{rl}[\Xi, w, \tau]}(x^t)} \;\leq\; m \ln k - \ln \tau(\infty) - (m-1) \ln \tau\left(\frac{t_m}{m-1}\right). \quad (3.21)$$

Proof. Fix a switch sequence σ^{t-1} with $m-1$ occurrences of s at positions $t_2, \ldots, t_m$, and let $t_1 = 0$. For $j = 1, \ldots, m-1$, let $\delta_j = t_{j+1} - t_j$ denote the length of block j. From the definition of the interpolator

above, we obtain

$$-\ln \pi_{\mathrm{RL}[\tau]}(\sigma^{t-1}) \;=\; -\ln \tau(z \geq t - t_m) - \sum_{j=1}^{m-1} \ln \tau(\delta_j) \;\leq\; -\ln \tau(\infty) - \sum_{j=1}^{m-1} \ln \tau(\delta_j).$$

Since $-\ln \tau$ is concave, by Jensen's inequality we have

$$\sum_{j=1}^{m-1} \frac{-\ln \tau(\delta_j)}{m-1} \leq -\ln \tau\left(\sum_{j=1}^{m-1} \frac{\delta_j}{m-1}\right) = -\ln \tau\left(\frac{t_m}{m-1}\right).$$

In other words, the block lengths δ_i are all equal in the worst case. Combining this with Corollary 3.5.4 we obtain the result. □

We have seen that the run-length model reduces to fixed share if the prior on switch distances τ is geometric, so that it can be evaluated in linear time in that case. We also obtain a linear time algorithm when τ has finite support, because then only a constant number of states can receive positive weight at any sample size. For this reason it can be advantageous to choose a τ with finite support, even if one expects that arbitrarily long distances between consecutive switches may occur. Expert sequences with such longer distances between switches can still be represented with a truncated τ using a sequence of reflexive switches from and to the same expert. This way, long runs of the same expert receive exponentially small, but positive, probability.

To compare the performance of the run-length model to the bound (3.16) for fixed share, assume $t_m = t - 1$ and define τ as in (3.20). The bound (3.21) becomes

$$m \ln k - \ln \tau(\infty) + (m-1)\left(\ln \frac{t-1}{m-1} + 2 \ln \ln \left(\frac{t-1}{m-1} + e\right) + e\right)$$

To maximise the difference, we use the optimising parameter α^* for fixed share, and we lower bound the entropy using (3.15). The gap between the bounds is then given by

$$-\ln \tau(\infty) + 2(m-1) \ln \ln \left(\frac{t-1}{m-1} + e\right) + (m-1)e.$$

At this modest price, the run-length model does not require tuning any parameters, its regret depends on t_m instead of t, and it may take advantage of clustered switches, although this is not expressed by Theorem 3.5.9.

3.5.6 Ordered Experts

In the models discussed so far, once a switch occurs, it is equally easy to switch to any of the available experts, as $\mathbb{B}_s$ prescribes uniform redistribution of the probability mass. This approach is reasonable if we do not know anything about the relationship between the experts; furthermore it has the advantage that percolating probabilities through $\mathbb{B}_s$ requires only $O(k)$ operations, while we would need $O(k^2)$ operations to support arbitrary transition probabilities between the experts. In this section we consider an interesting alternative that both makes intuitive sense and allows for efficient computation.

Assume that the experts can be sensibly organised using a line or ring topology, with the interpretation that switches between two experts are more likely if they are close together on this structure than if they are far apart. As an example, in a density estimation problem, one may define experts to estimate the underlying density of the data using a histogram model with 1, 2, ... bins respectively. In this case it is clear that, typically, the optimal number of bins to use increases gradually as more observations are gathered, so switching from a 10-bin histogram to a 11-bin histogram is more likely than, say, switching to a 1,000-bin histogram.

We will simplify matters further by postulating that the probability of a switch between any pair of experts who are δ apart is the same. Furthermore, for simplicity of exposition we identify the experts with the integers, $\Xi = \mathbb{Z}$. (In practice it is of course not possible to work with an infinite set of experts, but this can be resolved by simply changing the forward algorithm to drop all probability mass that at any time becomes propagated to an expert outside of the considered range.)

Now the notion of similarity between experts may be expressed by a kernel, i.e. a probability distribution on distances. The distribution on experts at time $t + 1$ is the *convolution* of the kernel with the distribution at time t. For kernel κ and distribution λ (both either discrete or

continuous), the convolution $\kappa * \lambda$ is defined by

$$(\kappa * \lambda)(x) \;\coloneqq\; \mathrm{E}_{\delta\sim\kappa}\left[\lambda(x-\delta)\right].$$

This approach can be lifted to the level of states: sometimes it may be sensible to order all states involved in an expert HMM. However, for simplicity we will consider the interpolating model of Section 3.5.2, where the transitions of $\mathbb{B}_{\mathsf{s}}$ are replaced by a convolution κ on the experts. The HMM implementing these convolutions is $\textsc{kernel}[\kappa] \coloneqq \langle Q, Q_{\mathrm{p}}, \mathrm{P}_{\circ}, \mathrm{P}_{\rightarrow}, \Lambda\rangle$, defined as follows

$$\begin{aligned} Q = Q_{\mathrm{p}} &= \mathbb{Z}\times\mathbb{Z}_{+} & \Lambda(\xi,t) &= \xi \\ \mathrm{P}_{\circ}(\xi,1) &= \kappa(\xi) & \mathrm{P}(\langle\xi,t\rangle \to \langle\xi',t+1\rangle) &= \kappa(\xi'-\xi). \end{aligned}$$

For this scenario, we derive the following analogue of Corollary 3.5.4:

3.5.10. Corollary (Kernel Interpolation Regret). *Let* $\mathbb{B}_{\mathsf{n}} = \textsc{bayes}[\mathbb{Z},\kappa]$ *and* $\mathbb{B}_{\mathsf{s}} = \textsc{kernel}[\kappa]$. *Fix* ξ^t. *Set* $\sigma_i = \mathsf{s}$ *iff* $\xi_{i+1} \neq \xi_i$, *and for* $1 \le j \le m$ *let* k_j *denote the expert used in the* j*th block. Further let* $k_0 = 0$. *Then for all* x^t*:*

$$\ln \frac{P_{\xi^t}(x^t)}{P_{\mathbb{B}_{\mathsf{n}}\otimes_{\mathfrak{H}}\mathbb{B}_{\mathsf{s}}}(x^t)} \;\le\; -\ln \pi_{\mathfrak{H}}(\sigma^{t-1}) - \sum_{j=1}^{m} \ln \kappa(k_j - k_{j-1}).$$

Proof. As before, we identify productive states and experts to get

$$\pi_{\textsc{bayes}[\mathbb{Z},\kappa]}(\xi_1) = \pi_{\textsc{kernel}[\kappa]}(\xi_1) = \kappa(\xi_1),$$
$$\pi_{\textsc{bayes}[\mathbb{Z},\kappa]}(\xi_i = \xi_{i-1}|\xi_{i-1}) = 1,$$

and

$$\pi_{\textsc{kernel}[\kappa]}(\xi_i|\xi_{i-1}) = \kappa(\xi_i - \xi_{i-1}).$$

Now Lemma 3.5.3 yields $\pi_{\mathbb{B}_{\mathsf{n}}\otimes_{\mathfrak{H}}\mathbb{B}_{\mathsf{s}}}(\xi^t) \;\ge\; \pi_{\mathfrak{H}}(\sigma^{t-1}) \;\prod_{j=1}^{m} \kappa(k_j - k_{j-1})$, and the result follows by (3.10). □

From the Convolution Theorem, we know that any convolution $\kappa * \lambda$ on k experts can be carried out in $O(k \log k)$ time using the Fast Fourier Transform algorithm, see e.g. [32, 21]. Thus, the ordered expert approach, which is in fact orthogonal to all the interpolating models described in previous sections, seems to provide a very attractive tradeoff between time complexity and expressive power.

In the following we consider a particular kernel for which the convolution can be performed in $O(k)$ time using a much simpler algorithm. It also has an interesting interpretation as a nice model for "parameter drift".

3.5.7 Parameter Drift

So far, we have discussed strategies where we follow a Bayesian prediction strategy which is interrupted every now and then by switching events. This is reflected by the regret bound Corollary 3.5.10, which consists of a term for the cost of specifying the indices of the switches, and a second term for the cost of specifying which experts are involved in the switches.

In this section we take a radically different approach. Rather than thinking of sporadic abrupt changes in the relative predictive performance of the experts, we now imagine that their performance changes gradually over time. Sticking to the ordered experts approach, as before we identify the set of experts with the integers, $\Xi = \mathbb{Z}$. However, in this section we will bound the regret in terms of the total amount of *drift* in ξ^t:

$$d \;=\; \sum_{i=1}^{t} |\delta_i|, \qquad \text{where} \qquad \delta_1 = \xi_1 \quad \text{and} \quad \delta_i = \xi_i - \xi_{i-1} \text{ for } 1 < i \leq t,$$

which can be viewed as the length of the path described by ξ^t.

As an example, one may consider the switching model proposed by Monteleoni and Jaakkola (see Section 3.5.4.2). They essentially instantiate a number of fixed share models, for various values of the switching rate α. These fixed share instances are prediction strategies, and can therefore be interpreted as experts themselves. However, it seems reasonable to assume that in many cases the optimal switching rate α might be subject to drift: it might vary somewhat as time progresses. Therefore it may be beneficial to combine these "fixed share experts" using a model that can represent parameter drift. The resulting loss can be bounded in terms of the amount of drift that occurs in the reference sequence of switching parameters. For parameter drift we no longer use an interpolation, as in previous sections, because switches no longer have special status. Instead, shifts between experts are possible at each time step, through convolution with the following kernel, parameterised by $0 < \alpha < 1$:

$$\kappa_\alpha(\delta) \;:=\; \alpha^{|\delta|} \frac{1-\alpha}{1+\alpha}.$$

This kernel can be implemented with the HMM $\textsc{kernel}[\kappa_\alpha]$ from the previous section, but as it turns out it is possible to represent the same

kernel using a different HMM $\textsc{pd}[\alpha]$, defined in Figure 3.10, that uses silent states to reduce the number of edges, allowing the convolution to be carried out in time proportional to the number of experts considered.

3.5.11. Theorem (Parameter Drift Regret). *Fix any data x^t and reference sequence ξ^t with total drift d. Let $H(P,Q) = -\sum_x P(x) \ln Q(x)$ denote the cross entropy. Then*

$$\ln \frac{P_{\xi^t}(x^t)}{P_{\textsc{pd}[\alpha]}(x^t)} \;\leq\; t\,\mathrm{H}(\kappa_{\alpha^*}, \kappa_\alpha) \;=\; -t \ln \frac{1-\alpha}{1+\alpha} - d \ln \alpha,$$

where $\alpha^* = \operatorname{argmax}_\alpha \pi_{\textsc{pd}[\alpha]}(\xi^t) = \sqrt{1 + (t/d)^2} - (t/d)$.

Proof. Applying Lemma 3.4.1, the left-hand side is bounded above by $-\ln \pi_{\textsc{pd}[\alpha]}(\xi^t)$. Since $\{\kappa_\alpha\}$ is an exponential family with unit carrier,

$$-\ln \pi_{\textsc{pd}[\alpha]}(\xi^t) \;=\; -\ln \prod_{i=1}^{t} \kappa_\alpha(\delta_i) \;=\; t\,\mathrm{E}_{\kappa_{\alpha^*}}\left[-\ln \kappa_\alpha(\delta)\right] \;=\; t\,\mathrm{H}(\kappa_{\alpha^*}, \kappa_\alpha).$$

The right equality follows from

$$\pi_{\textsc{pd}[\alpha]}(\xi^t) \;=\; \prod_{i=1}^{t} \kappa_\alpha(\delta_i) \;=\; \alpha^d \left(\frac{1-\alpha}{1+\alpha}\right)^t.$$

The parameter α^* that maximises the likelihood of ξ^t is found by equating the derivative to zero. □

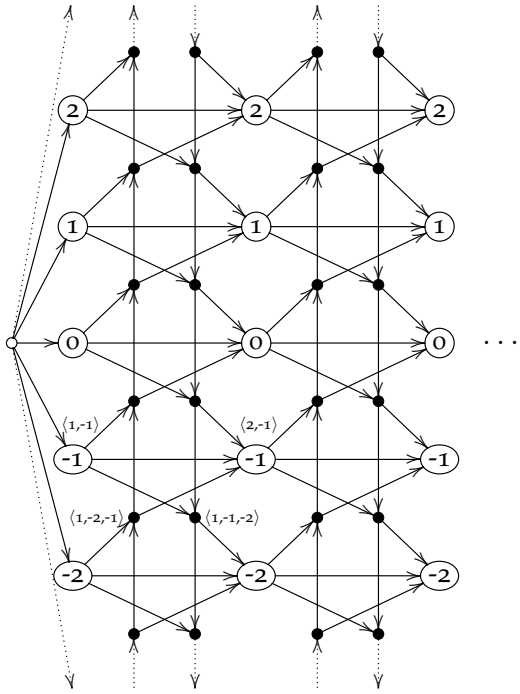
We can be somewhat more precise about how much it can hurt performance to use a suboptimal parameter α. The following theorem, which bounds the regret with respect to the optimal parameter-drift model, is an analogue of Equation 3.14 for fixed share. The theorem applies to a wide class of kernel HMMs, but in particular it holds for the parameter-drift model $\textsc{pd}[\alpha]$.

3.5.12. Theorem (Kernel ML Regret). *Fix a sequence of outcomes x^t and let $\hat{\eta} = \operatorname{argmax}_\eta P_{\textsc{kernel}[\kappa_\eta]}(x^t)$ for some exponential family $\{\kappa_\eta\}$. We have*

$$\ln \frac{P_{\textsc{kernel}[\kappa_{\hat{\eta}}]}(x^t)}{P_{\textsc{kernel}[\kappa_\eta]}(x^t)} \;\leq\; t\,\mathrm{D}\left(\kappa_{\hat{\eta}} \middle\| \kappa_\eta\right)$$

Proof. Since the transition probabilities associated with each productive state (i.e. the kernel κ_η) are an exponential family distribution, we can apply Lemma 3.4.3 with $Q^\dagger$ equal to the set of all productive states. □

Figure 3.10 Parameter drift: PD[α]



$$\mathrm{PD}[\alpha] = \langle Q, Q_{\mathrm{p}}, \mathrm{P}_{\circ}, \mathrm{P}_{\rightarrow}, \Lambda \rangle \qquad Q = Q_{\mathrm{s}} \cup Q_{\mathrm{p}}$$

$$Q_{\mathrm{p}} = \mathbb{Z}_+ \times \mathbb{Z} \quad \mathrm{P}_{\circ}(\langle 1, \xi \rangle) = \kappa_{\alpha}(\xi) \quad \Lambda(t, \xi) = \xi$$

$$Q_{\mathrm{s}} = \mathbb{Z}_+ \times \{\langle i, i+1 \rangle, \langle i, i-1 \rangle \mid i \in \mathbb{Z}\}$$

$$\mathrm{P}\begin{pmatrix} \langle t, \xi{-}1, \xi \rangle \to \langle t, \xi, \xi{+}1 \rangle \\ \langle t, \xi{+}1, \xi \rangle \to \langle t, \xi, \xi{-}1 \rangle \\ \langle t, \xi{-}1, \xi \rangle \to \langle t{+}1, \xi \rangle \\ \langle t, \xi{+}1, \xi \rangle \to \langle t{+}1, \xi \rangle \\ \langle t, \xi \rangle \to \langle t{+}1, \xi \rangle \\ \langle t, \xi \rangle \to \langle t, \xi, \xi{+}1 \rangle \\ \langle t, \xi \rangle \to \langle t, \xi, \xi{-}1 \rangle \end{pmatrix} = \begin{pmatrix} \alpha \\ \alpha \\ 1-\alpha \\ 1-\alpha \\ (1{-}\alpha)/(1{+}\alpha) \\ \alpha/(1+\alpha) \\ \alpha/(1+\alpha) \end{pmatrix}$$

3.5.7.1 Getting rid of α

The parameter drift model as discussed so far shares both the elegance of the fixed share algorithm and its awkward dependence on a parameter α. However, most of the techniques to avoid specifying α that were discussed in previous sections can be adapted to the parameter drift model. We will not discuss all these in detail, but consider only an adaptation of the trick that we used in Section 3.5.3. Namely, we let the kernel parameter α decrease with time.

3.5.13. THEOREM (Decreasing Drift Regret). *Let P_{PD} denote the ES-joint based on the parameter drift model with time-dependent kernel κ_{α_i} with $\alpha_i = 1/(i+1)$. For any data x^t and reference sequence ξ^t with total drift d, we have*

$$\ln \frac{P_{\xi^t}(x^t)}{P_{\text{PD}}(x^t)} \;\leq\; (d+2)\ln(t+1).$$

Proof. We first expand

$$\pi_{\text{PD}}(\xi^t) \;=\; \prod_{i=1}^{t} \kappa_{\alpha_i}(\delta_i) \;=\; \prod_{i=1}^{t} \alpha_i^{|\delta_i|}\frac{1-\alpha_i}{1+\alpha_i} \;=\; \prod_{i=1}^{t}(i+1)^{-|\delta_i|}\frac{i}{i+2} \;=$$
$$\frac{2}{(t+1)(t+2)}\prod_{i=1}^{t}(i+1)^{-|\delta_i|}.$$

For fixed total drift d, it is clear that this probability is minimised by $|\delta_i| = 0$ for $1 \leq i < t$ and $|\delta_t| = d$. Therefore

$$\pi_{\text{PD}}(\xi^t) \geq \frac{2}{(t+1)(t+2)}(t+1)^{-d} \geq (t+1)^{-d-2}.$$

We now take the $-\ln$ and apply Lemma 3.4.1 to complete the proof. □

3.5.8 White-Box Experts

So far, we have considered various interpolations, but we have always used a Bayesian mixture for $\mathbb{B}_n$. Thus we interpreted normal operation (no switch) as sticking to the same expert. Another interpretation, introduced in Chapter 4, instantiates $\mathbb{B}_n$ with an HMM that is able to learn some pattern of the data, e.g. the trend of the samples. Then, under normal operation, we keep on learning this trend. For $\mathbb{B}_s$ we

choose the HMM that collects the weights, and redistributes according to the initial distribution of $\mathbb{B}_n$, fulfilling the same role as EM took for BAYES. Thus, on a switch, everything is forgotten and learning the trend restarts from scratch. See Chapter 4 for examples and analysis of regret and running time for the FS$[\alpha]$ interpolator. The analyses can easily be adapted to the other interpolators described above.

3.6 Extensions

In this section we describe a number of such extensions to the framework described above. In Section 3.6.1 we outline a possible generalisation of the considered class of HMMs, allowing the ES-prior to depend on observed data. In Section 3.6.2 we try to find out which expert was best at a particular time step. In Section 3.6.3 we explain a minor modification of the algorithm that will disallow switching from an expert to that same expert. Finally in Section 3.6.4 we indicate how our approach can be generalised to work with any mixable loss function.

3.6.1 Data-Dependent Priors

When we discussed using HMMs to define ES-priors we imposed the restriction that for each state the associated Ξ-PFS should be independent of the previously produced experts. Indeed, conditioning on the *expert history* would increase the running time dramatically as all possible histories would have to be considered. However, conditioning on the *past observations* can be done *at no additional cost*, as the data are *observed*. Using this freedom would typically require additional knowledge about the process being modelled, in violation of our slogan "we do not understand the data". However, we may also condition on some function of the data that does not require too much domain specific knowledge to interpret. An interesting case is obtained by conditioning on the vector of losses (cumulative or incremental) incurred by the experts. This extends expressive power: the resulting ES-joints are generally not decomposable into an ES-prior and expert PFSs. An example is the Variable Share algorithm introduced in [80].

3.6.2 Expert Estimation

The forward algorithm computes the probability of the data, that is

$$P(x^t) \;=\; \sum_{q^\lambda : q_\lambda \in Q_{\{t\}}} P(x^t, q^\lambda).$$

Instead of the entire sum, we are sometimes interested in the sequence of states q^λ that contributes most to it:

$$\operatorname*{argmax}_{q^\lambda} P(x^t, q^\lambda) \;=\; \operatorname*{argmax}_{q^\lambda} P(x^t|q^\lambda)\pi(q^\lambda).$$

The Viterbi algorithm [146] is used to compute the most likely sequence of states for HMMs. It can be easily adapted to handle silent states. However, we may also write

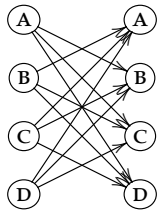
$$P(x^t) \;=\; \sum_{\xi^t} P(x^t, \xi^t),$$

and wonder about the sequence of *experts* ξ^t that contributes most. This problem is harder because several states can produce the same expert simultaneously (i.e. in the same $Q_{\{t\}}$); in other words a single sequence of experts can be generated by many different sequences of states. So we cannot use the Viterbi algorithm as it is. The Viterbi algorithm can be extended to compute the MAP expert sequence for general HMMs, but the resulting running time explodes. Still, the MAP ξ^t can be sometimes be obtained efficiently by exploiting the structure of the HMM at hand. This turns out to be possible for the switch distribution; the algorithm is given in [101].

As an alternative way to gain insight, one may run the forward and backward algorithms to compute $P(x^i, q_i^{\mathrm{p}})$ and $P(x^t|q_i^{\mathrm{p}}, x^i)$. Recall that q_i^{p} is the productive state that is used at time i. From these we can compute the a posteriori probability $P(q_i^{\mathrm{p}}|x^t)$ of each productive state q_i^{p}. That is, the posterior probability taking all the available data into account (including observations that were made later than time i). This is a standard way to analyse data in the HMM literature, see e.g. [146]. We can then project the posterior on *states* down to obtain the posterior probability $P(\xi_i|x^t)$ of each *expert* $\xi_i \in \Xi$ at each time $i = 1, \ldots, t$. This gives us a sequence of mixture weights over the experts that we can, for example, plot on a $\Xi \times t$ grid. On the one hand this gives us a mixture

Figure 3.11 Irreflexive switching

(a) State transition diagram



(b) Transition probabilities

$$\begin{aligned} \mathrm{P}(\xi \to \xi') &= w(\xi' | \xi' \neq \xi) \\ &= \frac{w(\xi')}{1 - w(\xi)} \end{aligned}$$

(c) Linear-time implementation

Input: Weights $a_1, \ldots, a_k$ at time t
Output: Weights $b_1, \ldots, b_k$ at time $t+1$
$p \leftarrow \sum_{i=1}^{k} \frac{a_i}{1-w(i)}$
for $i = 1 \ldots, k$ **do**
  $b_i \leftarrow w(i)\big(p - \frac{a_i}{1-w(i)}\big)$
end for

over experts for each time instance, obviously a richer representation than just single experts. On the other hand we lose the temporal correlations that can be important in MAP calculation, as each time instance is treated separately.

3.6.3 Omitting Reflexive Switches

In most of the models that we present, we allow reflexive switches, and prove a regret bound with a term of the form $m \ln k$. By disallowing reflexive switches (i.e. switches from and to the same expert), this regret bound can be sharpened to $\ln k + (m-1)\ln(k-1)$. One $\ln k$ term remains, since the expert in the first block has no predecessor. This can be done by using the HMM shown in Figure 3.11a for the switching behaviour $\mathbb{B}_{\mathbf{s}}$ instead of $\textsc{em}[\Xi, w]$. The transition probabilities are shown in Figure 3.11b. For the uniform prior they all reduce to $1/(k-1)$.

Note that the state transition diagram has $O(k^2)$ edges. Still, due to its regular structure, weights can be propagated forward in time $O(k)$ using the algorithm shown in Figure 3.11c.

3.6.4 Mixable Loss Functions

We presented log-loss regret bounds for experts that sequentially produce probability distributions on the next outcome. Not all prediction tasks are in this form, for example, we may be asked to make a point prediction based on real-valued expert advice and be scored using quadratic loss. Fortunately, several loss functions are *mixable* [25, 75], in that for each mixture of predictions, there is a single prediction whose loss is always less than the exponentiated average loss. Mixable losses include log loss, quadratic loss, Hellinger loss and entropic loss. 0/1 loss and absolute loss are not mixable.

Prediction strategies that are obtained by running the forward algorithm on any HMM can be adapted to mixable losses straightforwardly, by preprocessing the input to and post-processing the output of the forward algorithm for sequential prediction. On the input side, expert predictions are transformed into probabilities. On the output side, the posterior distribution on the next expert (3.3) is transformed (using the mixability condition) into a single prediction. The resulting prediction strategy has the *same* mixable-loss regret bound as the original prediction strategy (although possibly expressed in different units). The details of the reduction can be found in Section 4.6.

3.7 Conclusion

In prediction with expert advice, we have at our disposal a number of sequential prediction strategies ("experts"), and the goal is to combine their predictions into a single prediction, by taking a weighted mixture at every time step.

We take the Bayesian approach and model such combinations by defining prior distributions on expert sequences (ES-priors). The (infinitely long) expert sequence defines which expert is used at which time. Prediction then amounts to "integrating out" those experts in the sequence that are used at other time steps than the one predicted. The challenge is to identify those models that provide good tradeoffs between predictive performance and time complexity.

We employ hidden Markov models (HMMs) to specify ES-priors, since their explicit representation of the current state and state-to-state evolution naturally fit the temporal correlations we seek to model. For

reasons of efficiency we use HMMs with silent states. The standard algorithms for HMMs (Forward, Backward, Viterbi and Baum-Welch) can be used to answer questions about the ES-prior as well as the induced distribution on data. The running time of the forward algorithm can be read off directly from the graphical representation of the HMM.

Our approach allows unification of many existing expert models. We focus on models for *tracking the best expert*, where the loss incurred by a prediction strategy is compared to the loss incurred if the data are optimally divided into m blocks, and the best expert is used within each block. The discrepancy ("regret") is then bounded in terms of variables such as the current time t, the number of experts k, and the number of blocks m. In each case, we recover (sometimes improve) both the regret bound and the running time known from the literature.

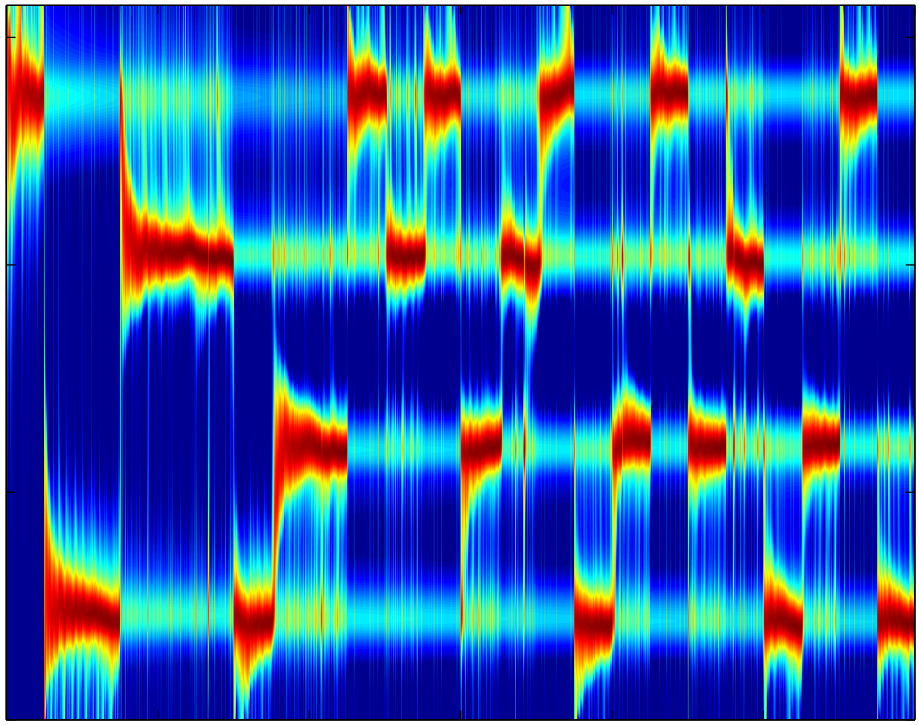
We use our unifying framework not only to succinctly summarise and contrast many key algorithms from the literature, but also to describe a number of new models. In particular the models with decreasing probability of switching (Section 3.5.3), the run-length model (Section 3.5.5) and the models that assume the experts to be ordered (Section 3.5.6) are new, are computationally efficient and have competitive regret bounds.

Acknowledgements

Peter Grünwald's and Tim van Erven's suggestions significantly improved this chapter. Thanks also go to Mark Herbster for an enjoyable afternoon exchanging ideas, which has certainly influenced the shape of this chapter. We thank Wojciech Kotłowski for proofreading.

Chapter 4

Freezing & Sleeping



Abstract A problem posed by Freund is how to efficiently track a small pool of experts out of a much larger set. This problem was solved when Bousquet and Warmuth introduced their mixing past posteriors (MPP) algorithm in 2001.

In Freund's problem the experts would normally be considered black boxes. However, in this chapter we re-examine Freund's problem in case the experts have internal structure that enables them to learn. In this case the problem has two possible interpretations: should the experts learn from all data or only from the subsequence on which they are being tracked? The MPP algorithm solves the first case. Our contribution is to generalise MPP to address the second option. The results we obtain apply to any expert structure that can be formalised using (expert) hidden Markov models. Curiously enough, for our interpretation there are *two* natural reference schemes: freezing and sleeping. For each scheme, we provide an efficient prediction strategy and prove the relevant loss bound.

4.1 Introduction

Freund's problem arises in the context of prediction with expert advice [25]. In this setting a sequence of outcomes needs to be predicted, one outcome at a time. Thus, prediction proceeds in rounds: in each round we first consult a set of experts, who give us their predictions. Then we make our own prediction and incur some loss based on the discrepancy between this prediction and the actual outcome. The goal is to minimise the difference between our cumulative loss and some reference scheme. For this reference there are several options; we may, for example, compare ourselves to the cumulative loss of the best expert in hindsight. A more ambitious reference scheme was proposed by Yoav Freund in 2000.

Freund's Problem Freund asked for an efficient prediction strategy that suffers small additional loss compared to the following reference scheme:

(a) Partition the data into several subsequences.
(b) Select an expert for each subsequence.
(c) Sum the loss of the selected experts on their subsequences.

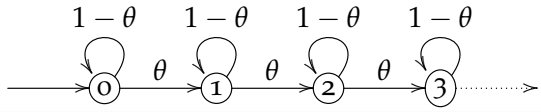
In 2001, Freund's problem was addressed by Bousquet and Warmuth, who developed the efficient algorithm called mixing past posteriors (MPP) [19]. MPP's loss is bounded by the loss of Freund's scheme plus some overhead that depends on the number of bits required to encode the partition of the data, and it has found successful application in [70]. Problem solved. Or is it?

4.1.1 Three Reference Schemes

In this paper we take another look at Freund's reference scheme for *learning experts* and ask: if an expert is selected for some segment, then should the expert learn from all data or only from the data in that segment?

We may assume that the experts do not know the segmentation chosen in step a of the reference scheme. (Otherwise, why wouldn't we just ask them?) Hence if we treat the experts as black boxes and only ask for their prediction at each time step as in [19], it is natural that they

Figure 4.1 Example learning expert $\mathsf{DM}[\theta]$, which learns a drifting mean, specified by its state transition diagram.



learn from all data. We call this interpretation of Freund's problem the *full reference scheme*.

However, as the following example will illustrate, it may be beneficial if experts learn only from the segment for which they are selected, because they may get confused by data in other segments that follow a different pattern. As a slight complication, it will turn out that we have a further choice: whether to tell a learning expert the timing of its segment or not, which generally makes a difference. When segment timing is preserved, we obtain the *sleeping reference scheme*; when segment timing is *not* preserved we obtain the *freezing reference scheme*. The next intuitive example demonstrates that the full, freezing and sleeping reference schemes are fundamentally different, and that the latter two can be dramatically more appropriate for prediction with learning experts.

4.1.1.1 Motivating Example: Drifting Mean

In applications one would usually build up complicated prediction strategies from simpler ones in a hierarchical fashion. Following that fashion, we first define simple constant experts, parametrised by $\mu \in \mathbb{R}$, which predict according to a normal distribution with mean μ and unit variance in each round.

Learning Experts Now define a learning expert $\mathsf{DM}[\theta]$, as displayed in Figure 4.1, that has a stochastic model for the (unobservable) drift of μ over time. This *drifting mean* learning expert predicts according to a hidden Markov model in which the hidden state at time t is μ_t and the production probability of an outcome given μ_t is determined by the simple expert with parameter μ_t. Initially, $\mu_1 = 0$ with probability one. Then $\mu_{t+1} = \mu_t + 1$ with probability θ and $\mu_{t+1} = \mu_t$ with probability $1 - \theta$ for some fixed parameter θ.

The expert $\mathsf{DM}[\theta]$ may be said to be learning, because its posterior distribution of μ_t given outcomes $x_1, \ldots, x_{t-1}$ indicates how much credibility the expert assigns to each value of μ_t: high weight on, say, $\mu_t = 3$ indicates that $\mathsf{DM}[\theta]$ considers it likely for $\mu_t = 3$ to give the best prediction for x_t.

Data Consider the two artificial data sets displayed in Figures 4.2a and 4.2b. These data sets were obtained as follows. First, we generated two straight-line data sets, with outcomes increasing at a rate of 0.1 and 0.3 per trial respectively. Then we divided both data sets in segments of 100 outcomes each. The data in Figure 4.2a were obtained by *interleaving* 10 segments from the 0.1 and 0.3 data sets, whereas the data in Figure 4.2b were obtained by *alternating* 10 segments from the 0.1 and 0.3 data sets. By construction, the freezing reference scheme is suited for the data in Figure 4.2a, while the sleeping reference scheme is appropriate for the data in Figure 4.2b.

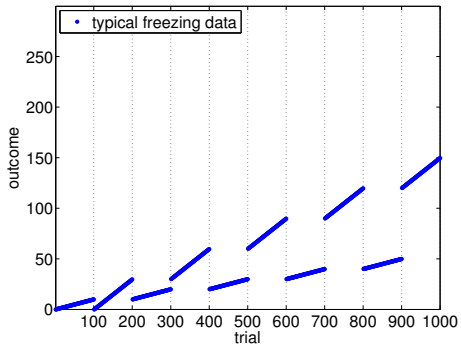
Prediction Task We now evaluate the performance of the three reference schemes on the two data sets. In each case we consider two experts: $\mathsf{DM}[0.1]$ and $\mathsf{DM}[0.3]$, and split the data into two subsequences (step (a), according to the true rate, either 0.1 or 0.3. We predict all outcomes for which the actual rate was $\theta \in \{0.1, 0.3\}$ using the expert $\mathsf{DM}[\theta]$.

The difference between the three schemes lies in which data is used by both experts to learn from. In the full reference scheme $\mathsf{DM}[0.1]$ and $\mathsf{DM}[0.3]$ are shown all the data, even those samples they do not predict. In the two other reference schemes, on the other hand, $\mathsf{DM}[0.1]$ only sees the data for which it is selected, that is, the data with true rate 0.1. Similarly, $\mathsf{DM}[0.3]$ only sees the data with true rate 0.3. For freezing $\mathsf{DM}[\theta]$ predicts as if the data it has observed are the only data, thus the original timing of the samples is lost. For sleeping the original timing of the samples is preserved, and $\mathsf{DM}[\theta]$ has to predict with uncertainty about the intermediate unobserved samples.

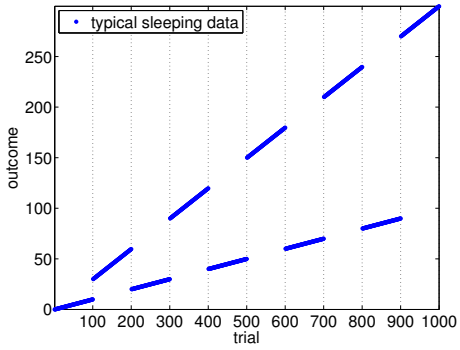
Posteriors Figures 4.2c and 4.2d show the posterior distribution of the expert $\mathsf{DM}[0.1]$ on states after 200 trials for each reference scheme. These posterior distributions can be interpreted as the belief of the learning expert $\mathsf{DM}[0.1]$ about the unobserved drifting mean after 200 trials.

Figure 4.2 The difference between the full, freezing and sleeping reference schemes. Note the logarithmic scale of the y-axis in (e) and (f)!

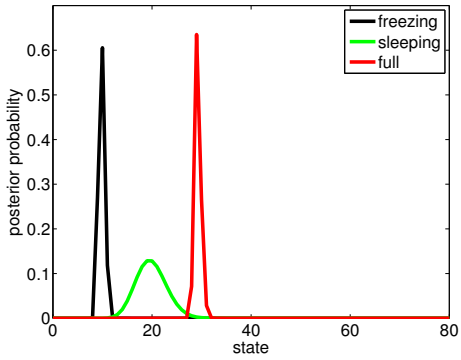
(a) Suitable freezing data



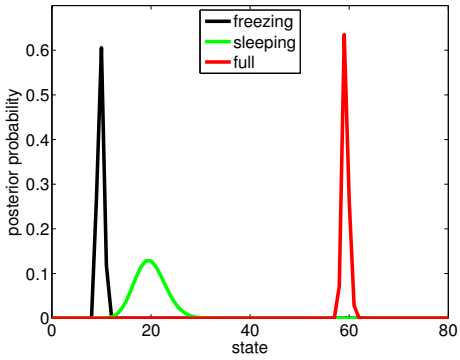
(b) Suitable sleeping data



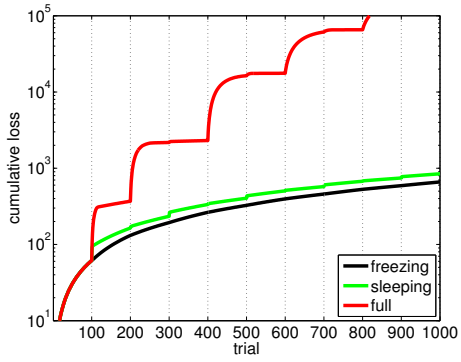
(c) Belief of DM[0.1] after 200 trials of (a)



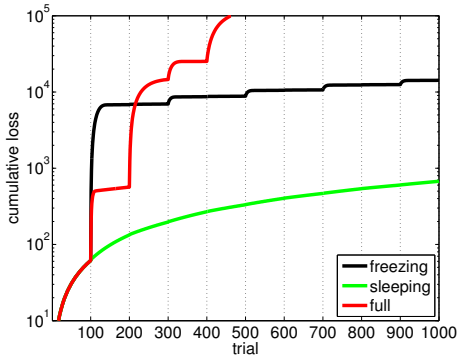
(d) Belief of DM[0.1] after 200 trials of (b)



(e) Cumulative loss on data (a)



(f) Cumulative loss on data (b)



We see in Figure 4.2c that, for the freezing data, the expert posterior obtained by the freezing reference scheme matches the 201st outcome (which is 10 in the freezing data set) best. Recall that this posterior is obtained by first showing $\mathsf{DM}[0.1]$ outcomes 1 through 100, and then asking it to predict outcome 201 as if it was the next outcome in the sequence.

We also see in Figure 4.2d that, for the sleeping data, the expert posterior obtained by the sleeping reference scheme matches the 201st outcome (which is 20 in the sleeping data set) best. Recall that this posterior is obtained by first showing $\mathsf{DM}[0.1]$ outcomes 1 through 100, and then asking it to predict outcome 201 with all intermediate outcomes unobserved.

Finally, we see that in both cases, the expert posterior obtained by the full reference scheme, which shows *all* outcomes to $\mathsf{DM}[0.1]$, overshoots: the expert is confused by observing the intermediate outcomes.

Loss These snapshots of the expert's posteriors provide an intuitive understanding of what the reference schemes do and which one is appropriate. We now quantify the predictive performance by looking at the resulting cumulative loss. Figures 4.2e and 4.2f show the cumulative log(arithmic) loss for all three reference schemes. Note that the difference between the schemes is so large that their losses had to be plotted on a logarithmic scale.

We see in Figure 4.2f that for the sleeping data the sleeping reference scheme has much smaller loss than the other two schemes. And for the freezing data the freezing reference scheme has the smallest loss by far, as shown in Figure 4.2e. (Mind the logarithmic scale of the y-axis, which puts the loss of sleeping deceptively close to the loss of freezing in Figure 4.2e: a constant offset indicates a fixed multiplicative overhead.) In both cases the reason for the large differences between the reference schemes is that both experts $\mathsf{DM}[0.1]$ and $\mathsf{DM}[0.3]$ get confused if they learn from the wrong data.

Note that for this synthetic example, we knew which partitioning into subsequences to choose, since we constructed the data ourselves. For real data a partitioning is not readily available. The challenge addressed in this chapter is to *learn* the best partition of the data online.

4.1.1.2 Structured Experts

In this chapter, we solve Freund's problem under the interpretation that experts only observe the subsequence on which they are evaluated. Of course, for *arbitrary* experts, this is impossible. For in the setting of prediction with expert advice (see [25]), the expert predictions that we receive each round are *always* in the context of all data. We have no access to the experts' predictions in the context of any subsequence, and these predictions may differ drastically from those on the whole data.

Often however, experts have internal structure. For example, in [108, 80, 180, 181] adaptive prediction strategies (i.e. learning experts) are explicitly constructed from basic experts. To represent such structured experts, we use the general framework called *expert hidden Markov models* (EHMMs), that was introduced in Chapter 3. EHMMs are hidden Markov models in which the production probabilities are determined by expert advice. A structured expert in EHMM form provides sufficient information about its predictions on any isolated subsequence.

Many strategies for prediction with expert advice (i.e. learning experts) can be rendered as EHMMs. For example all adaptive strategies in the papers above (see Chapter 3). But there are also strategies that cannot be brought into EHMM form, like e.g. *follow the perturbed leader* [73] and *variable share* [80].

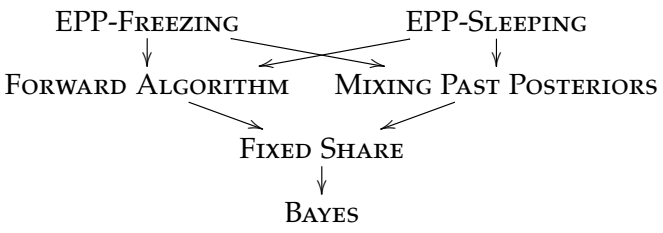
Our approach may also be of interest to machine learning with regular hidden Markov models (HMMs) [146]. Although existing approaches to shift between multiple HMMs [65, 66, 104] usually focus on change-point detection, prediction seems a highly related issue.

4.1.2 Overview

After preliminaries we start by reviewing the main existing loss bound for mixing past posteriors in Section 4.3. Then, in Section 4.4, we review EHMMs as a way to represent structured experts.

The next section, Section 4.5, contains our results for Freund's problem when structured experts are evaluated on isolated subsequences. We formalise sleeping and freezing as two different ways of presenting a subsequence of the data to an EHMM, and present the *evolving past posteriors* (EPP) algorithm that takes an EHMM as input. The EPP algorithm has two variants, which both generalise the mixing past pos-

Figure 4.3 Generalisation relation among prediction strategies



teriors algorithm in a different way: EPP-Sleeping for sleeping and EPP-Freezing for freezing. The relation between EPP and other existing prediction strategies is shown in Figure 4.3. There $A \to B$ means that by carefully choosing prediction strategy A's parameters it reduces to strategy B.

In order to understand EPP, we verify that it produces the same predictions for any two EHMMs that are equivalent in an appropriate sense, and analyse its running time. We then proceed to show our main result, which is that the losses of EPP-Freezing and EPP-Sleeping are bounded by the loss of their appropriate reference scheme plus a complexity penalty that depends on the number of bits required to encode the reference partition in the same way as for mixing past posteriors. In fact, our bounds (slightly) improve the known loss bound for mixing past posteriors. Thus we solve Freund's problem with learning experts presented as EHMMs, both for freezing and for sleeping.

We first derive our results only for logarithmic loss. This allows us to use familiar concepts and results from probability theory and refer to the interpretation of log loss as a codelength [25]. In Section 4.6 we conclude by proving that any algorithm that satisfies certain weak conditions, in particular EPP, directly generalises to an algorithm for arbitrary mixable losses with the appropriate loss bounds.

4.2 Preliminaries

Prediction With Expert Advice Each round t, we first receive advice from each expert $e \in \mathcal{E}$ in the form of an action $a_t^e \in \mathcal{A}$. Then we distill our own action $a_t^{\mathsf{alg}} \in \mathcal{A}$ from the expert advice. Finally, the actual outcome $x_t \in \mathcal{X}$ is observed, and everybody suffers loss as specified

by a fixed loss function $\ell\colon \mathcal{A} \times \mathcal{X} \to [0, \infty]$. Thus, the performance of a sequence of actions $a_1 \cdots a_T$ upon data $x_1 \cdots x_T$ is measured by the cumulative loss $\sum_{t=1}^{T} \ell(a_t, x_t)$.

Log Loss For *log loss* the actions $\mathcal{A}$ are probability distributions on $\mathcal{X}$ and $\ell(\mathrm{p}, x) = -\log \mathrm{p}(x)$, where log denotes the natural logarithm. It is important to notice that minimising log loss is equivalent to maximising the predicted probability of outcome x. We write p_t^e for the prediction of expert e at time t and denote these predictions jointly by $\mathrm{p}_t^{\mathcal{E}}$.

Subsequences For $m \leq n$, we abbreviate $\{m, \ldots, n\}$ to $m{:}n$. For completeness, we set $m{:}n = \emptyset$ for $m > n$. For any sequence $y_1, y_2, \ldots$ and any set of indices $\mathcal{C} = \{i_1, i_2, \ldots\}$ we write $y_{\mathcal{C}}$ for the subsequence $\langle y_i \rangle_{i \in \mathcal{C}}$. For example, $x_{\mathcal{C}} = \langle x_i \rangle_{i \in \mathcal{C}}$ and $\mathrm{p}_{1:T}^{\mathcal{E}} = \mathrm{p}_1^{\mathcal{E}}, \ldots, \mathrm{p}_T^{\mathcal{E}}$. If members of a family $\mathbb{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots\}$ are pairwise disjoint and together cover $1{:}T$ ($\bigcup \mathbb{C} = 1{:}T$), then we call $\mathbb{C}$ a *partition* of $1{:}T$, and its members *cells*.

4.3 Mixing Past Posteriors

Mixing past posteriors (MPP) is a strategy for prediction with expert advice. It operates by maintaining a table of so-called posterior distributions on the set of experts. Each round, we first compute the predictive distribution on experts by mixing all the posteriors in the table. Then the next outcome is predicted by mixing the expert predictions according to this distribution. Finally, the next outcome is observed. The predictive distribution on experts is conditioned on this outcome, and the posterior distribution thus obtained is appended to the table of posteriors. Note the recursive construction of the distributions in the table; they are not Bayesian posteriors, but conditioned mixtures of all earlier distributions from that same table.

We will not formally introduce MPP here, but recover it as a special case of both the freezing and sleeping algorithms in Section 4.5.4. Here we state the classical loss bound [19, Theorem 7], introducing our notation along the way. This loss bound relates the loss of MPP to Freund's full reference scheme, where we choose a partition of the data (step a) and select an expert for each partition cell (step b). We measure expert

performance (step c) using the predictions issued in the context of all data, i.e. the full interpretation of Freund's scheme.

4.3.1 Loss Bound

We bound the overhead of MPP over the full reference scheme in terms of the complexity of the reference partition. We first state the theorem, and then explain the ingredients. We write $\mathrm{P}_w^{\mathsf{MPP}}(x_{1:T})$ for the probability that MPP assigns to data $x_{1:T}$ (so $-\log\bigl(\mathrm{P}_w^{\mathsf{MPP}}(x_{1:T})\bigr)$ is MPP's cumulative log loss).

4.3.1. Theorem ([19, Theorem 7]). *For any mixing scheme β, Bayesian joint distribution P^{B} with prior distribution w on experts, partition $\mathbb{C}$ of $1{:}T$, data $x_{1:T}$ and expert predictions $\mathrm{p}^{\mathcal{E}}_{1:T}$*

$$\mathrm{P}_w^{\mathsf{MPP}}(x_{1:T}) \;\geq\; \beta(\mathbb{C})\mathrm{P}^{\mathsf{B}}_{\mathbb{C}}(x_{1:T}). \tag{4.1}$$

A *mixing scheme* β is a sequence $\beta_1, \beta_2, \ldots$ of distributions, where β_{j+1} is a probability distribution on $0{:}j$. In [19] several mixing schemes are listed, e.g. *Uniform Past* and *Decaying Past*. A mixing scheme is turned into a distribution on partitions as follows. Let $\mathbb{C}$ be a partition of $1{:}T$, and let $i \in 1{:}T$. *The cell of i*, denoted $\mathbb{C}(i)$, is the unique $\mathcal{C} \in \mathbb{C}$ such that $i \in \mathcal{C}$. We write $\mathrm{prev}^{\mathbb{C}}(i)$ for *the predecessor of i*, defined as the largest element in $\mathbb{C}(i) \cup \{0\}$ that is smaller than i. Using this notation, the distribution on partitions is given by

$$\beta(\mathbb{C}) \coloneqq \prod_{t \in 1:T} \beta_t(\mathrm{prev}^{\mathbb{C}}(t)).$$

Note that this distribution is potentially *defective*; two elements $i < j$ cannot share the same nonzero predecessor, but β_i may assign nonzero probability to $\mathrm{prev}^{\mathbb{C}}(j)$ nonetheless.

Now that we have seen how the loss bound encodes partition, we turn to $\mathrm{P}^{\mathsf{B}}_{\mathbb{C}}(x_{1:T})$, the probability of the data $x_{1:T}$ given a particular partition $\mathbb{C}$. To compute it, we treat the cells independently (4.2), and per cell we use the Bayesian mixture with prior w on experts (4.3), thus

mixing the predictions the experts issued in the context of all data (4.4).

$$\mathrm{P}^{\mathsf{B}}_{\mathbb{C}}(x_{1:T}) := \prod_{\mathcal{C}\in\mathbb{C}} \mathrm{P}^{\mathsf{B}}_{\mathbb{C}}(x_{\mathcal{C}}), \text{ where} \tag{4.2}$$

$$\mathrm{P}^{\mathsf{B}}_{\mathbb{C}}(x_{\mathcal{C}}) := \sum_{e\in\mathcal{E}} w(e)\mathrm{p}^{e}_{\mathcal{C}}(x_{\mathcal{C}}) \text{ and} \tag{4.3}$$

$$\mathrm{p}^{e}_{\mathcal{C}}(x_{\mathcal{C}}) := \prod_{i\in\mathcal{C}} \mathrm{p}^{e}_{i}(x_i). \tag{4.4}$$

A second bounding step allows us to relate the performance of MPP directly to Freund's full scheme. Let w be the uniform prior over a finite set of experts $\mathcal{E}$, and select an expert $e^{\mathcal{C}}$ for each partition cell $\mathcal{C} \in \mathbb{C}$. Then bound each sum (4.3) from below by one of its terms to obtain

4.3.2. Corollary. $$\mathrm{P}^{\mathsf{MPP}}_{w}(x_{1:T}) \;\geq\; \beta(\mathbb{C})\,|\mathcal{E}|^{-|\mathbb{C}|} \prod_{\mathcal{C}\in\mathbb{C}} \mathrm{p}^{e^{\mathcal{C}}}_{\mathcal{C}}(x_{\mathcal{C}}).$$

Thus the log-loss overhead of MPP over the full reference scheme is bounded by $-\log\beta(\mathbb{C}) + |\mathbb{C}|\log|\mathcal{E}|$, which can be related to the number of bits to encode the chosen partition and the selected experts for each cell [19].

Convex Combinations In [19], the authors make a point of selecting a *convex combination of experts* for each subsequence, where the loss of a convex combination of experts is the weighted average *loss* of the experts. The loss of such a convex combination is therefore *always* higher than the loss of its best expert. Uniform bounds in terms of arbitrary experts, like Corollary 4.3.2, apply in particular to the best expert, and hence to any convex combination. Therefore, without loss of generality, we do not discuss convex combinations any further.

Interpreting Freund's Problem The loss bound Theorem 4.3.1 shows that MPP solves the black-box-experts interpretation of Freund's problem. This can be seen clearly in (4.4). To predict the subsequence $x_{\mathcal{C}}$, it uses predictions $\mathrm{p}^{e}_{\mathcal{C}}$ which were issued in the context of all data. This means that the experts observe the entire history $x_{1:i}$ before predicting the next outcome X_{i+1}.

Switching between *learning* experts that observe all data is useful when the data are homogeneous, and the experts learn its global pattern at different speeds. In such cases we want to train each expert on

all observations, for then by switching at the right time, we can predict each outcome using the expert that has learned most *until then*. This scenario is analysed in [178], where experts are parameter estimators for a series of statistical models of increasing complexity.

On the other hand, if the data have local patterns then our new interpretation of Freund's problem applies, and we want to train each expert on the subsequence on which it is evaluated, so that it can exploit its local patterns. To solve Freund's problem for such learning experts, we need to know about its internal structure.

4.4 Structured Experts

Assume there is only a single expert and fix a reference partition. Suppose we want to predict as if the expert is restarted on each cell of the partition, when in reality the expert just makes her predictions as if all the data were in a single cell. Then clearly this is impossible if we treat this expert completely as a black box: if we do not know what the expert's predictions would have been if a certain outcome were, say, the start of a new cell, then we cannot match these predictions.

The expert therefore needs to reveal to us some of her internal state. To this end, we will represent the parts of her internal state that will *not* be revealed to us by lower level experts that we will treat as black boxes, and assume our main expert combines the predictions of these base experts using an *expert hidden Markov model* (EHMM).

4.4.1 EHMMs

Expert Hidden Markov Models (EHMMs) were introduced in Chapter 3 as a language to specify strategies for prediction with expert advice. We briefly review them here. An *EHMM* $\mathcal{A}$ is a probability distribution that is constructed according to the Bayesian network in Figure 4.4. It is used to sequentially predict outcomes $X_1, X_2, \ldots$ which take values in outcome space $\mathcal{X}$. At each time t, the distribution of X_t depends on a hidden state Q_t, which determines mixing weights for the experts' predictions. Formally, the *production function* $\mathrm{p}_\downarrow$ determines the interpretation of a state: it maps any state $q_t \in \mathcal{Q}$ to a distribution $\mathrm{p}_\downarrow^{q_t}$ on the identity E_t of the expert that should be used to predict X_t. Then given $E_t = e$, the distribution of X_t is base expert e's prediction p_t^e. It remains

Figure 4.4 Bayesian network specification of an EHMM

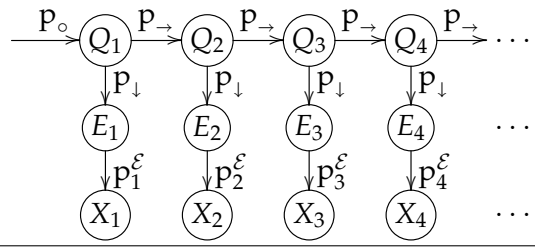
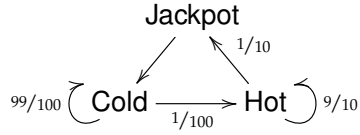


Figure 4.5 Hidden state transitions in slot machine HMM



to define the distribution of the hidden states. The starting state Q_1 has *initial distribution* $p_\circ$, and the state evolves according to the *transition function* $p_\rightarrow$, which maps any state q_t to a distribution $p_\rightarrow^{q_t}$ on states.

An EHMM $\mathcal{A}$ defines a prediction strategy as follows; after observing $x_{1:t}$, predict outcome X_{t+1} using the marginal $\mathcal{A}(X_{t+1}|x_{1:t})$, which is a *mixture* of the expert's predictions $p_{t+1}^{\mathcal{E}}$.

4.4.1. Example (Any Ordinary HMM). To illustrate how ordinary hidden Markov models are a special case of EHMMs, consider the following naive gambler's HMM model of an old-fashioned slot machine: in each round the gambler inserts one nickel into the slot machine and then the machine pays out a certain number of nickels depending on its hidden internal state: in state Cold it pays out nothing; in state Hot it pays out an amount between one and five nickels, uniformly at random; and then there's Jackpot in which it always pays out ten nickels. The machine always starts in state Cold and the state transitions are as in Figure 4.5.

To make an EHMM out of this HMM, we just identify experts with states: $\mathcal{Q} = \mathcal{E} = \{\mathsf{Cold}, \mathsf{Hot}, \mathsf{Jackpot}\}$, $p_\downarrow^e(e) = 1$, and each expert predicts according to the corresponding payout scheme. The distributions on states follow the original HMM: $p_\circ(\mathsf{Cold}) = 1$ and $p_\rightarrow$ as in Figure 4.5. ◇

4.4.2. Example (Bayes on base experts). We identify the Bayesian distribution with prior w on base experts $\mathcal{E}$ and the EHMM with $\mathcal{Q} = \mathcal{E}$, $\mathrm{p}_\circ = w$, and $\mathrm{p}^e_\to(e) = \mathrm{p}^e_\downarrow(e) = 1$, since their marginals coincide. Despite its deceptive simplicity, this EHMM *learns*: its marginal distribution on the next outcome is a mixture of the expert's predictions according to the Bayesian posterior. ◇

4.4.3. Example (Bayes on EHMMs). Fix EHMMs $\mathcal{A}^1, \ldots, \mathcal{A}^n$ with disjoint state spaces and the same basic experts, and let w be a prior distribution on $1{:}n$. The Bayesian mixture EHMM has state space $\mathcal{Q} = \bigcup_i \mathcal{Q}^i$, and for any two states $q, q' \in \mathcal{Q}^i$ belonging to the same original EHMM, $\mathrm{p}_\circ(q) = w(i)\,\mathrm{p}^i_\circ(q)$, $\mathrm{p}^q_\to(q') = \mathrm{p}^{i,q}_\to(q')$ and $\mathrm{p}^q_\downarrow(e) = \mathrm{p}^{i,q}_\downarrow(e)$. Again, this EHMM *learns* which of the given EHMMs is the best predictor. ◇

4.4.2 The Forward Algorithm

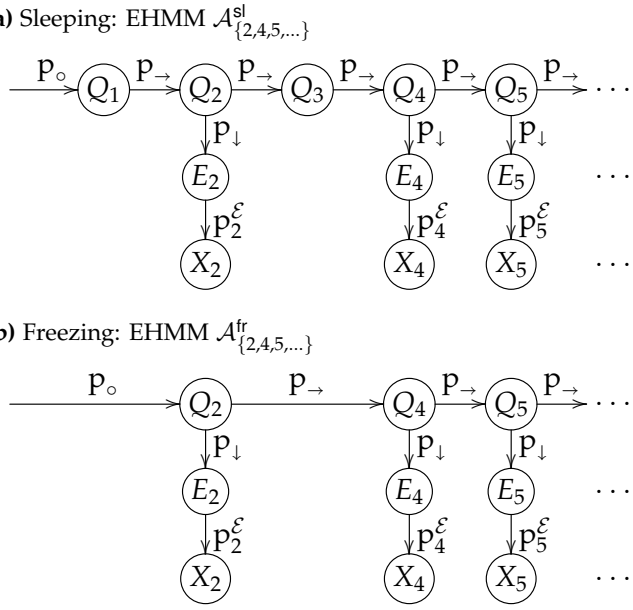
Sequential predictions for EHMMs can be computed efficiently using the *forward algorithm* (see Algorithm 3.1 on page 77), which maintains a posterior distribution over states, and predicts each outcome with a mixture of the experts' predictions. Given a posterior $\lambda_t(Q_t) = \mathcal{A}(Q_t | x_{1:t-1})$ for the hidden state at time t, the forward algorithm predicts x_t using the marginal of $\mathcal{A}(Q_t, E_t, X_t | x_{1:t-1})$. Then, after observing outcome x_t, it updates its posterior λ_t for Q_t to a posterior λ_{t+1} for Q_{t+1}.

For finite $\mathcal{Q}$, $\mathcal{E}$ and $\mathcal{X}$, the running time of the algorithm is determined by this last posterior update step, which in general may require $O\big(|\mathcal{Q}|^2\big)$ computation steps for each round t. On T outcomes, this gives a total running time of $O\big(|\mathcal{Q}|^2 \cdot T\big)$. In Appendix 4.A we provide a more careful analysis.

4.5 Freezing & Sleeping

Let $x_{1:T} = x_1, \ldots, x_T$ be a sequence of data and suppose that a reference partition $\mathbb{C}$ of $1{:}T$ is given in advance. We are interested in the performance of a structured expert $\mathcal{A}_\mathbb{C}$, which for each cell $C \in \mathbb{C}$ runs a separate instance of the structured expert $\mathcal{A}$ on the subsequence x_C. This leaves unspecified, however, whether the original timing of x_C should be preserved when x_C is presented to $\mathcal{A}$. This is a modelling choice,

Figure 4.6 Sleeping and Freezing EHMMs on outcomes $x_{\{2,4,5,\dots\}}$



which depends on the application at hand. We therefore treat both the case where the timing is preserved, which we call *sleeping*, and the case where the timing is not preserved, which we call *freezing*. (See also Figure 4.2 in the introduction.)

Sleeping We say that the instance of $\mathcal{A}$ that is used to predict cell $\mathcal{C}$ is sleeping if it does notice the passing of time during outcomes outside of $\mathcal{C}$, even though it does not observe them. We write $\mathcal{A}^{\mathsf{sl}}_{\mathcal{C}}$ for the resulting EHMM, which is shown in Figure 4.6a for the example $\mathcal{C} = \{2, 4, 5, \dots\}$. Notice that $\mathcal{A}^{\mathsf{sl}}_{\mathcal{C}}$ contains all five states $Q_{1:5}$, even though it does not observe x_1 or x_3. This has the effect that state transitions from e.g. Q_2 to Q_4 are composed of two transition steps according to $\mathrm{p}_\rightarrow$. The distributions on individual cells combine into the following distribution on all data $x_{1:T}$:

$$\mathcal{A}^{\mathsf{sl}}_{\mathbf{C}}(x_{1:T}) := \prod_{\mathcal{C}\in\mathbf{C}} \mathcal{A}^{\mathsf{sl}}_{\mathcal{C}}(x_{\mathcal{C}}).$$

To memorise the nature of sleeping, one may think of the way television channels get interleaved as you zap between them: a channel not being

watched is not paused, but instead continues broadcasting even when its content is not observed.

Freezing In freezing, the instance of $\mathcal{A}$ that is used to predict cell $\mathcal{C} \in \mathbb{C}$ is frozen when outcomes outside of $\mathcal{C}$ occur: its internal state should not change based on those outcomes. (Of course we have no control over the base experts on which $\mathcal{A}$ is based, so they may do whatever they please with such data. We therefore do have to preserve the timing of the base experts' predictions.) The resulting EHMM $\mathcal{A}^{\mathsf{fr}}_{\mathcal{C}}$ is shown for the example $\mathcal{C} = \{2, 4, 5, \ldots\}$ in Figure 4.6b. Note that Q_2, Q_4 and Q_5 are the first, second and third state of $\mathcal{A}^{\mathsf{fr}}_{\mathcal{C}}$; state transitions between them consist of a single transition step according to $\mathrm{p}_{\rightarrow}$. The resulting distribution on all data is defined by

$$\mathcal{A}^{\mathsf{fr}}_{\mathbb{C}}(x_{1:T}) := \prod_{\mathcal{C} \in \mathbb{C}} \mathcal{A}^{\mathsf{fr}}_{\mathcal{C}}(x_{\mathcal{C}}).$$

One might associate freezing with the way different e-mail conversations get interleaved in your inbox (if it is sorted by order of message arrival): a conversation about your latest research is paused (remains frozen) regardless of how much spam you receive in between.

4.5.1 An Infeasible Solution

The freezing or sleeping distributions can be computed if the reference partition $\mathbb{C}$ is given in advance. The problem we are addressing, however, is that we do not assume $\mathbb{C}$ to be known. An easy (but impractical) solution to this problem is to predict according to the Bayesian mixture of all possible partitions: let w be a prior on the set of all possible partitions and predict such that the joint distribution on all data is given by

$$\mathfrak{B}(x) := \sum_{\mathbb{C}} w(\mathbb{C}) \mathcal{A}^{\mathsf{f/s}}_{\mathbb{C}}(x),$$

where $\mathsf{f/s}$ denotes either fr for freezing or sl for sleeping. Lower bounding the sum by the term for the reference partition $\mathbb{C}$ directly gives an upper bound on the log loss:

$$-\log \mathfrak{B}(x) \leq -\log w(\mathbb{C}) - \log \mathcal{A}^{\mathsf{f/s}}_{\mathbb{C}}(x).$$

To predict according to $\mathfrak{B}$ in general would require an exponential amount of state to keep track of all possible partitions, which is completely impractical. In the following section we therefore present generalisations to both sleeping and freezing of the mixing past posteriors algorithm and show that their running time is comparable to that of the forward algorithm on $\mathcal{A}$ itself. Then in section Section 4.5.3 we prove bounds that relate the additional loss to the encoding cost of the reference partition $\mathbb{C}$.

4.5.2 The EPP Algorithm

Here we present a generalisation of the mixing past posteriors (MPP) algorithm, which we call *evolving past posteriors* (EPP). It is based on the view that MPP internally uses the Bayesian mixture of base experts, which is a standard EHMM. Given this perspective and after making the distinction between sleeping and freezing, the generalisation to other EHMMs is straightforward. We will discuss the connections between MPP and EPP in more detail in Section 4.5.4.

The EPP algorithm has variants for sleeping and freezing, which are both given in Algorithm 4.1. It takes an EHMM $\mathcal{A}$ and mixing scheme β (see Section 4.3.1) as input. Given a distribution λ_t on the hidden state Q_t at time t, the EPP algorithm predicts X_t exactly like the forward algorithm. It differs from the forward algorithm, however, in the way it computes λ_t. Whereas in the forward algorithm λ_t may be interpreted as the posterior distribution on Q_t, in the EPP algorithm λ_t is a β-mixture of *the algorithm's own past posteriors*. This recursive nature of EPP, which it inherits from the MPP algorithm, makes it hard to analyse.

We denote by $\mathrm{P}^{\mathsf{fr}}_{\mathcal{A}}$ and $\mathrm{P}^{\mathsf{sl}}_{\mathcal{A}}$ the probability distributions on random variables $\langle Q_t, E_t, X_t \rangle_{t \in \mathbb{N}}$ defined by EPP-Freezing and EPP-Sleeping on EHMM $\mathcal{A}$ and mixing scheme β. For both $\mathsf{f/s} \in \{\mathsf{sl}, \mathsf{fr}\}$

$$\mathrm{P}^{\mathsf{f/s}}_{\mathcal{A}}(q_{1:T}, e_{1:T}, x_{1:T}) \;=\; \prod_{t \in 1:T} \mathrm{p}^{\mathsf{alg}}_t(q_t, e_t, x_t).$$

4.5.2.1 Representation Invariance

Let $\mathcal{A}^1$ and $\mathcal{A}^2$ be EHMMs that are based on the same set of experts $\mathcal{E}$, but have different state spaces. We call $\mathcal{A}^1$ and $\mathcal{A}^2$ *equivalent* if

Algorithm 4.1 Evolving past posteriors (EPP)

Input:

- An EHMM $\mathcal{A}$ with components $\mathrm{p}_\circ$, $\mathrm{p}_\rightarrow$ and $\mathrm{p}_\downarrow$ (see Section 4.4)
- A mixing scheme $\beta_1, \beta_2, \ldots$ (see Section 4.3.1 and Section 4.5.2.2)
- Expert predictions $\mathrm{p}_1^{\mathcal{E}}, \mathrm{p}_2^{\mathcal{E}}, \ldots$ and data $x_1, x_2, \ldots$

Output: Predictions $\mathrm{p}_1^{\mathsf{alg}}, \mathrm{p}_2^{\mathsf{alg}}, \ldots$
Storage: Past posteriors $\pi_1, \pi_2, \ldots$ on $\mathcal{Q}$, the states of $\mathcal{A}$

Algorithm

1: Set the first posterior to the initial distribution of $\mathcal{A}$

$$\pi_1(q_1) \leftarrow \mathrm{p}_\circ(q_1)$$

2: **for** $t = 1, 2, \ldots$ **do**

3: Form λ_t, the current configuration, as the β_t-mixture of past posteriors:

$$\lambda_t(q_t) \leftarrow \sum_{0 \le j < t} \beta_t(j) \pi_{j+1}(q_t).$$

4: Compute $\mathrm{p}_t^{\mathsf{alg}}$, the joint distribution on states, experts and outcomes:

$$\mathrm{p}_t^{\mathsf{alg}}(q_t, e_t, x_t) \leftarrow \lambda_t(q_t)\, \mathrm{p}_\downarrow^{q_t}(e_t) \mathrm{p}_t^{e_t}(x_t).$$

5: Predict x_t using the marginal $\mathrm{p}_t^{\mathsf{alg}}(x_t)$,

6: Observe x_t. Suffer log loss

$$\ell_t^{\mathsf{alg}} \leftarrow -\log\bigl(\mathrm{p}_t^{\mathsf{alg}}(x_t)\bigr).$$

7: Perform loss update and state evolution to obtain the next posterior

$$\pi_{t+1}(q_{t+1}) \leftarrow \sum_{q_t \in \mathcal{Q}} \mathrm{p}_t^{\mathsf{alg}}(q_t | x_t)\, \mathrm{p}_\rightarrow^{q_t}(q_{t+1}).$$

8: Only for sleeping: perform state evolution for all past posteriors ($1 \le j \le t$)

$$\pi_j(q_{t+1}) \leftarrow \sum_{q_t \in \mathcal{Q}} \pi_j(q_t)\, \mathrm{p}_\rightarrow^{q_t}(q_{t+1}).$$

9: **end for**

Table 4.1 Mixing schemes

Mixing scheme	$\beta_{t+1}(t)$	$\beta_{t+1}(j)$ for $0 \leq j < t$
Yesterday	1	0
Fixed Share(α)	$1-\alpha$	α if $j = 0$ and 0 o.w.
Uniform past(α)	$1-\alpha$	α/t
Decaying past(α, γ)	$1-\alpha$	$\alpha(t-j)^{-\gamma}/Z_t$

$\mathcal{A}^1(e_{1:T}) = \mathcal{A}^2(e_{1:T})$ for all $e_{1:T}$. Consequently, equivalent EHMMs assign the same probability $\mathcal{A}^1(x_{1:T}) = \mathcal{A}^2(x_{1:T})$ to all data $x_{1:T}$, hence the difference between $\mathcal{A}^1$ and $\mathcal{A}^2$ is merely a matter of *representation*. As an important sanity check, we need to verify that EPP on either EHMM issues the same predictions.

4.5.1. Theorem (Invariance). *Let* f/s *denote either* fr *or* sl. *Fix equivalent EHMMs* $\mathcal{A}^1$ *and* $\mathcal{A}^2$. *Then for all data* $x_{1:T}$

$$\mathrm{P}^{\mathsf{f/s}}_{\mathcal{A}^1}(x_{1:T}) \;=\; \mathrm{P}^{\mathsf{f/s}}_{\mathcal{A}^2}(x_{1:T}).$$

Proof. Given in Appendix 4.C. □

Thus, from the perspective of predictive performance, the difference between $\mathcal{A}^1$ and $\mathcal{A}^2$ is irrelevant. Of course, it does matter for the computational cost of EPP, see Section 4.5.2.3.

4.5.2.2 Mixing Schemes

Bousquet and Warmuth [19] provide an extensive discussion of possible mixing schemes. Their loss bounds for various schemes carry over directly to our setting. It is interesting, however, to analyse the running times of the Fixed-Share to *uniform past* and to *decaying past* mixing schemes for EPP. For further information we refer the reader to [19].

Both schemes (see Table 4.1) depend on a *switching rate* $\alpha \in [0,1]$, which determines whether to continue with yesterday's posterior or switch back to an earlier one: $\beta_{t+1}(t) = 1-\alpha$ and $\sum_{0\leq j<t} \beta_{t+1}(j) = \alpha$.

Uniform Past Given the choice to switch back, the uniform past mixing scheme gives equal weights to the entire past: $\beta_{t+1}(j) = \alpha/t$ for $0 \leq j < t$.

Decaying Past The decaying past scheme assigns larger weight to the recent past: $\beta_{t+1}(j) = \alpha(t-j)^{-\gamma}/Z_t$ for $0 \leq j < t$, where $Z_t = \sum_{0 \leq j < t}(t-j)^{-\gamma}$ is a normalising constant and $\gamma \geq 0$ is a parameter that determines the rate of decay.

4.5.2.3 Running Times

Appendix 4.A provides a detailed comparison of the running times and space requirements of EPP and the forward algorithm. The upshot is that for the uniform past mixing scheme the sleeping variant of EPP is as efficient as the forward algorithm, in terms of both running time and space requirements; the freezing variant is equally efficient if the set of hidden states $\mathcal{Q}$ is finite, but may be a factor $O(T)$ less efficient on T outcomes for countably infinite $\mathcal{Q}$. The decaying past mixing scheme is a factor $O(T)$ less efficient (for both time and space) than uniform past in all cases, but may be approximated by a scheme described in [19] that reduces this factor to $O(\log T)$.

4.5.3 Loss Bound

We relate the performance of EPP-Freezing and EPP-Sleeping (defined in Algorithm 4.1) to that of $\mathcal{A}^{\mathsf{fr}}_{\mathbb{C}}$ and $\mathcal{A}^{\mathsf{sl}}_{\mathbb{C}}$ for all partitions $\mathbb{C}$ *jointly*.

4.5.2. Theorem (EPP Loss Bounds). *For both* $\mathsf{f/s} \in \{\mathsf{fr}, \mathsf{sl}\}$ *and any mixing scheme* β*, data* $x_{1:T}$ *and expert predictions* $\mathrm{p}^{\mathcal{E}}_{1:T}$

$$\mathrm{P}^{\mathsf{f/s}}_{\mathcal{A}}(x_{1:T}) \;\geq\; \sum_{\mathbb{C}} \beta(\mathbb{C}) \mathcal{A}^{\mathsf{f/s}}_{\mathbb{C}}(x_{1:T}). \tag{4.5}$$

Proof. Given in Appendix 4.B. □

Using this bound, we can relate the predictive performance of EPP-Sleeping and EPP-Freezing to that of $\mathcal{A}^{\mathsf{sl}}_{\mathbb{C}}$ and $\mathcal{A}^{\mathsf{fr}}_{\mathbb{C}}$ for any reference partition $\mathbb{C}$.

4.5.3. Corollary. $\mathrm{P}^{\mathsf{f/s}}_{\mathcal{A}}(x_{1:T}) \;\geq\; \beta(\mathbb{C}) \mathcal{A}^{\mathsf{f/s}}_{\mathbb{C}}(x_{1:T})$.

From the brutal way in which Corollary 4.5.3 was obtained, we may expect to often do much better in practice; *many* partitions may contribute significantly to (4.5).

4.5.4 Recovering MPP

We now substantiate our claim that EPP generalises MPP by proving that MPP results from running EPP-Freezing or EPP-Sleeping on the Bayesian EHMM (Example 4.4.2).

4.5.4. Theorem. *Let $\mathcal{A}$ be the Bayesian EHMM with initial distribution w, and let $\mathrm{P}_w^{\mathsf{MPP}}$ denote the probability distribution defined by MPP with prior w. Then for all data $x_{1:T}$*

$$\mathrm{P}_{\mathcal{A}}^{\mathsf{fr}}(x_{1:T}) \;=\; \mathrm{P}_{\mathcal{A}}^{\mathsf{sl}}(x_{1:T}) \;=\; \mathrm{P}_w^{\mathsf{MPP}}(x_{1:T}).$$

Proof. The difference between freezing and sleeping (line 8) evaporates since state evolution is the identity operation. By identifying states and experts the MPP algorithm [19, Figure 1] remains. □

The theorem does not require the set of experts $\mathcal{E}$ to be finite. If $\mathcal{E}$ is infinite (or too large), MPP is intractable. Still, a small EHMM may exist that implements Bayes (say with the uniform prior) on $\mathcal{E}$, and we can use EPP-Sleeping (which is faster than EPP-Freezing) for sequential prediction. For example, we may implement MPP on the infinite set of Bernoulli experts efficiently, in time $O(T^2)$, using EPP-Sleeping on the *universal element-wise mixture* EHMM of [100, §4.1].

4.5.4.1 Improved MPP Loss Bound

[19, Theorem 7] (our Theorem 4.3.1) bounds the overhead of MPP over Freund's full scheme in terms of $\beta(\mathbb{C})$, the complexity of the reference partition $\mathbb{C}$ according to the mixing scheme β. A more general bound follows directly from Theorems 4.5.2 and 4.5.4:

4.5.5. Corollary. $\quad \mathrm{P}_w^{\mathsf{MPP}}(x_{1:T}) \;\geq\; \sum_{\mathbb{C}} \beta(\mathbb{C}) \mathrm{P}_{\mathbb{C}}^{\mathsf{B}}(x_{1:T}).$

Even with a fixed reference partition $\mathbb{C}$ in mind, we get a better bound by considering small modifications of $\mathbb{C}$, e.g. finer partitions or partitions that disagree about a single round.

Adversarial Experts For each number of rounds T one can construct a set of T base experts and data $x_{1:T}$ such that the loss of Freund's full scheme is infinite for all partitions except the finest one. We simply

have expert t suffer infinite loss in all rounds other than t. In this pathological case the bounds in Theorem 4.3.1 for that partition and Corollary 4.5.5 are equal and tight.

4.5.4.2 Is EPP strictly more general than MPP?

A natural question is whether either EPP-Sleeping or EPP-Freezing can be implemented using MPP on a rich set of meta-experts. To preclude the trivial answer that regards either algorithm as a single meta-expert, we ask for a fixed construction that works for all mixing schemes.

Sleeping For any EHMM $\mathcal{A}$, EPP-Sleeping can be reduced to MPP on meta-experts. Let the set of meta-experts be $\mathcal{Q}^\infty$, the set of paths through the hidden states of $\mathcal{A}$. Each meta-expert $q_{\mathbb{N}}$ predicts x_t using the $\mathrm{p}_\downarrow^{q_t}$-mixture of base expert predictions. We set the prior w in MPP equal to the marginal probability measure of $\mathcal{A}$ on paths (as determined by $\mathrm{p}_\circ$ and $\mathrm{p}_\rightarrow$). We omit the proof that the predictions made by MPP on these meta-experts with prior w are equal to those made by EPP on $\mathcal{A}$.

Freezing The next example shows that EPP-Freezing really is more general than MPP. Fix two experts $\mathcal{E} = \{\mathrm{a}, \mathrm{b}\}$. Consider the EHMM $\mathcal{A}$ that predicts the first outcome using expert a, and the second outcome using expert b, i.e. $\mathcal{Q} = \mathcal{E}$, and $\mathrm{p}_\circ(\mathrm{a}) = \mathrm{p}_\rightarrow^{\mathrm{a}}(\mathrm{b}) = \mathrm{p}_\downarrow^{q}(q) = 1$. Running EPP-Freezing on $\mathcal{A}$ results in $\pi_2(\mathrm{b}) = \pi_1(\mathrm{a}) = 1$, so that the first outcome is predicted using expert a, and the second outcome is predicted using the β_2-mixture of experts. Thus any candidate meta-expert *must* predict the first outcome using base expert a. But that means that for MPP with prior w on meta-experts, the loss update has no effect, so that $w = \pi_1 = \pi_2 = \lambda_2$. Hence the second outcome will be predicted according to the prior mixture of experts. Since β_2 is arbitrary and w is fixed, there can be no general scheme to reduce EPP-Freezing to MPP.

4.6 Other Loss Functions

We will now show how the EPP algorithm for logarithmic loss can be directly translated into an algorithm with corresponding loss bound for any other mixable loss function. The same construction works for any

logarithmic loss algorithm that predicts according to a mixture of the experts' predictions at each trial and whose predictions only depend on the experts' past losses on outcomes that actually occurred.

Mixability A loss function $\ell\colon \mathcal{A} \times \mathcal{X} \to [0,\infty]$ is called *η-mixable* for $\eta > 0$ if any distribution p on experts $\mathcal{E}$ can be mapped to a single action $\mathrm{Pred}(p) \in \mathcal{A}$ in a way that guarantees that

$$\ell\big(\mathrm{Pred}(p), x\big) \le -\tfrac{1}{\eta} \log \mathbb{E}_{e\sim p}\Big[\exp\big(-\eta\ell(a^e, x)\big)\Big] \tag{4.6}$$

for all outcomes $x \in \mathcal{X}$ and expert predictions $a^{\mathcal{E}}$. It is called *mixable* if it is η-mixable for some $\eta > 0$ [25]. Mixability ensures that expert predictions for ℓ loss can be mixed in essentially the same way as for log loss.

For example, logarithmic loss itself is 1-mixable. And for $\mathcal{A} = [0,1]$ and $\mathcal{X} = \{0,1\}$ the *square loss* $\ell(a,x) := (a-x)^2$ is 2-mixable and the *Hellinger loss* $\ell(a,x) :=$
$((\sqrt{1-x} - \sqrt{1-a})^2 + (\sqrt{x} - \sqrt{a}))/2$ is $\sqrt{2}$-mixable.[75, 25]

The Benefits of Lying Given data $x_{1:t}$ and expert predictions $a^{\mathcal{E}}_{1:t}$, let $\ell^e_{1:t} := \ell(a^e_1, x_1), \ldots, \ell(a^e_t, x_t)$ denote the sequence of losses of expert e, and let $\ell^{\mathcal{E}}_{1:t}$ denote these losses jointly for all experts. In the special case that ℓ is the logarithmic loss we write $\ell\ell^e_{1:t}$ and $\ell\ell^{\mathcal{E}}_{1:t}$, respectively.

Suppose Alg is an algorithm for log loss that predicts each outcome x_t by mixing the experts' predictions $\mathrm{p}^{\mathcal{E}}_t$ according to the distribution $\mathrm{p}^{\mathsf{alg}}_t[x_{<t}, \ell\ell^{\mathcal{E}}_{<t}]$ on *experts*. The square-bracket expression indicates that $\mathrm{p}^{\mathsf{alg}}_t$ may depend on the past outcomes $x_{1:t-1}$ and the losses of the experts on these outcomes, but not on the experts' past or current predictions in any other way. Following this convention, the algorithm predicts x_t using:

$$\mathrm{p}^{\mathsf{alg}}_t[x_{<t}, \ell\ell^{\mathcal{E}}_{<t}](x_t) \;:=\; \sum_e \mathrm{p}^{\mathsf{alg}}_t[x_{<t}, \ell\ell^{\mathcal{E}}_{<t}](e)\,\mathrm{p}^e_t(x_t).$$

Now for any game with η-mixable loss ℓ and the same set of experts $\mathcal{E}$, we can derive from Alg an algorithm $\textsc{Alg}^{\eta}_{\ell}$ that predicts x_t according to

$$a_t^{\mathsf{alg}^{\eta}_{\ell}} \;:=\; \mathrm{Pred}\left(\mathrm{p}^{\mathsf{alg}}_t[x_{<t}, \eta\ell^{\mathcal{E}}_{<t}]\right).$$

Note that $\textsc{Alg}_\ell^\eta$ is lying to Alg: while Alg thinks it is playing a game for log loss in which experts have incurred log losses $\eta\ell^{\mathcal{E}}_{<t}$, in reality $\textsc{Alg}_\ell^\eta$ is playing a game for loss ℓ and is feeding Alg fake inputs and redirecting Alg's outputs. Let us now analyse the loss of the derived algorithm $\textsc{Alg}_\ell^\eta$.

4.6.1. Lemma (Other Loss Functions). *Suppose* Alg *is an algorithm for logarithmic loss that predicts according to* $\mathrm{p}^{\mathsf{alg}}_t[x_{<t}, \ell\ell^{\mathcal{E}}_{<t}]$ *at each time* t*,* ℓ *is an* η*-mixable loss function, and* $f(x_{1:T}, \ell^{\mathcal{E}}_{1:T})$ *is an arbitrary function that maps outcomes and expert losses to real numbers. Then any log loss bound for* Alg *of the form*

$$-\log \mathrm{P}^{\mathsf{alg}}(x_{1:T}) \;\le\; f(x_{1:T}, \ell\ell^{\mathcal{E}}_{1:T}) \qquad \text{for all } \mathrm{p}^{\mathcal{E}}_{1:T}, \tag{4.7}$$

directly implies the ℓ *loss bound for* $\textsc{Alg}_\ell^\eta$*:*

$$\ell(a^{\mathsf{alg}^\eta_\ell}_{1:T}, x_{1:T}) \;\le\; \tfrac{1}{\eta} f(x_{1:T}, \eta\ell^{\mathcal{E}}_{1:T}) \qquad \text{for all } a^{\mathcal{E}}_{1:T}. \tag{4.8}$$

Proof. Construct a log loss game in which at any time t each expert e predicts according to a distribution p^e_t such that $\mathrm{p}^e_t(x_t) = \exp(-\eta\ell^e_t)$ for the actual outcome x_t and p^e_t is arbitrary on other outcomes such that $\sum_{x_t} \mathrm{p}^e_t(x_t) = 1$. In this game the log loss of Alg is

$$-\log \mathrm{P}^{\mathsf{alg}}(x_{1:T}) = \sum_{t\in 1:T} -\log \mathrm{p}^{\mathsf{alg}}_t[x_{<t}, \eta\ell^{\mathcal{E}}_{<t}](x_t).$$

By η-mixability of ℓ

$$\begin{aligned}\ell(a^{\mathsf{alg}^\eta_\ell}_{1:T}, x_{1:T}) \;&=\; \sum_{t\in 1:T} \ell\Big(\operatorname{Pred}\big(\mathrm{p}^{\mathsf{alg}}_t[x_{<t}, \eta\ell^{\mathcal{E}}_{<t}]\big), x_t\Big) \\ &\le\; \tfrac{1}{\eta} \sum_{t\in 1:T} -\log \mathrm{p}^{\mathsf{alg}}_t[x_{<t}, \eta\ell^{\mathcal{E}}_{<t}](x_t). \end{aligned} \tag{4.9}$$

Combining with (4.7) and (4.9) completes the proof. □

Algorithms that satisfy the requirements of the lemma include Bayes, follow the (perturbed) leader, the forward algorithm, MPP and EPP. An algorithm that does not satisfy them is the last-step minimax algorithm [172], because it takes into account the experts' predictions on outcomes that do not occur.

In the literature it is common to construct algorithms for arbitrary mixable losses and point out their probabilistic interpretation for the special case of log loss [75, 80, 19]. Instead, we have proceeded the other way around: first we derived results for log loss and then we showed that they generalise to other losses. This allowed us to draw on concepts and results from probability theory like conditional probabilities, HMMs and the forward algorithm, without reproving them in a more general setting.

Lemma 4.6.1 generalises results by Vovk [183], who shows that the most important loss bounds for Bayes with logarithmic loss can actually also be derived for arbitrary mixable losses. Our algorithm ALG plays a role similar to his APA algorithm.

4.7 Discussion

Relearning vs Continuing to Learn Corollary 4.5.3 bounds the regret of EPP with respect to a reference partition $\mathbb{C}$ by $-\log \beta(\mathbb{C})$. Consider the asymptotic behaviour of this bound if $\mathbb{C}$ has infinitely many shifts. (A shift occurs when $\text{prev}^{\mathbb{C}}(t+1) \neq t$.) For both decaying past with $\gamma \leq 1$ (e.g. following recommendations in [19]) and uniform past (see Table 4.1) $\max_{0 \leq j < t} \beta_{t+1}(j)$ goes to zero as a function of t. Thus, the cost per shift (be it to continue an earlier cell or to start a new one) grows without bound. On the other hand for fixed share $\beta_{t+1}(0) = \alpha$ for all t, hence fixed share can start a new cell at fixed cost. It depends on the structured expert whether continuing previously selected cells at increasing cost is advantageous over relearning from scratch after each shift at fixed cost. For EHMM experts with a finite state space $\mathcal{Q}$ (including Bayes), relearning from scratch will cost at most a factor $|\mathcal{Q}|$ over learning on. This factor is constant, so that fixed share will eventually win.

4.8 Conclusion

We revisited Freund's problem, which asks for a strategy for prediction with expert advice that suffers small additional loss compared to Freund's reference scheme. We discussed the solution by Bousquet and Warmuth, which interprets the experts as black boxes. We proposed

a new interpretation of Freund's scheme which is natural for learning experts, namely to train experts on the subsequence on which they are evaluated. This allows the reference scheme to exploit local patterns in the data, and thus makes the problem harder.

We solved Freund's problem for structured experts that are represented as EHMMs, building on the work of Bousquet and Warmuth. We showed that our prediction strategies are efficient, and have desirable loss bounds that apply to all mixable losses.

4.A Running Times

We compare the running times on T outcomes of EPP and the forward algorithm, with respect to an arbitrary EHMM $\mathcal{A}$ with a countable set of hidden states $\mathcal{Q}$. For simplicity we assume that the sets of experts $\mathcal{E}$ and outcomes $\mathcal{X}$ are finite.

Let Q_t denote the hidden state of $\mathcal{A}$ at time t, and let $\mathrm{p}_\circ$, $\mathrm{p}_\rightarrow$ and $\mathrm{p}_\downarrow$ denote $\mathcal{A}$'s other components. Both algorithms base their predictions on a distribution λ_t on Q_t at time t, but differ in how they update λ_t after observing x_t. As the number of computations for this step depends on the size of the support of λ_t and on $\mathrm{p}_\rightarrow$, we will need the following concepts. For any probability distribution p on $\mathcal{Q}$, let $\mathrm{Sp}(\mathrm{p}) = \{q \in \mathcal{Q} \mid \mathrm{p}(q) > 0\}$ denote its support. We recursively define $\mathcal{Q}_t$, the set of states reachable in exactly t steps, and $\mathcal{Q}_{\leq t}$, the set of states reachable in at most t steps, by

$$\mathcal{Q}_1 := \mathrm{Sp}(\mathrm{p}_\circ), \qquad \mathcal{Q}_{t+1} := \bigcup_{q \in \mathcal{Q}_t} \mathrm{Sp}(\mathrm{p}_\rightarrow^q), \qquad \mathcal{Q}_{\leq t} := \bigcup_{i \in 1:t} \mathcal{Q}_i.$$

Obviously, $\mathcal{Q}_t \subseteq \mathcal{Q}_{\leq t} \subseteq \mathcal{Q}$ holds for all t. Let $g(S) := \sum_{q \in S} |\mathrm{Sp}(\mathrm{p}_\rightarrow^q)|$ be the number of outgoing transitions from any set of states $S \subseteq \mathcal{Q}$.

4.A.1 Forward

The forward algorithm computes λ_{t+1} by conditioning λ_t on x_t and applying the transition function $\mathrm{p}_\rightarrow$. As λ_t has support $\mathcal{Q}_t$, the forward algorithm requires $O\big(g(\mathcal{Q}_t)\big)$ work per time step, and $O\big(|\mathcal{Q}_t| + |\mathcal{Q}_{t+1}|\big)$ space. Notice that, for finite $\mathcal{Q}$, the number of transitions is bounded by $g(S) \leq |\mathcal{Q}|^2$ for any S. A rough upper bound on the total running

time of forward on T outcomes is therefore $O(|\mathcal{Q}|^2 T)$, which is linear in T.

4.A.2 EPP

The EPP algorithm comes in two variants: one for sleeping and one for freezing. For sleeping the order of the running time is determined by the evolution of past posteriors (line 8 in Algorithm 4.1); for freezing, which skips line 8, either computation of λ_t (line 3) or of the next posterior (line 7) is the dominant step. The main difference for the running times of the two variants, however, is that in sleeping π_j has support $\mathcal{Q}_t$ at any time t, whereas for freezing π_j has support $\mathcal{Q}_{\leq j}$.

4.A.2.1 Uniform Past

For the uniform past mixing scheme, one can keep track of $\sum_{j=0}^{t} \pi_j(q_t)$ to speed up computation of λ_{t+1}.

Sleeping This even works for sleeping, because applying the state evolution to this sum in line 8 of the algorithm is equivalent to applying it to the individual π_j and then summing. Consequently, sleeping requires $O(g(\mathcal{Q}_t))$ work and $O(|\mathcal{Q}_t| + |\mathcal{Q}_{t+1}|)$ space per time step, which makes it as efficient as the forward algorithm.

Freezing For freezing, computing the next posterior (line 7) determines the running time. It requires $O(g(\mathcal{Q}_{\leq t}))$ work and $O(|\mathcal{Q}_{\leq t+1}|)$ space per time step. Depending on the EHMM $\mathcal{A}$, this may be significantly slower than the forward algorithm. First, for finite $\mathcal{Q}$, each of $\mathcal{Q}_t$, $\mathcal{Q}_{\leq t}$ and $\mathcal{Q}$ have size $O(1)$ in t, and freezing runs in time $O(T)$, just like the forward algorithm. Second, for infinite $\mathcal{Q}$, $\mathcal{Q}_{\leq t}$ may be unbounded as a function of t. Still, on T outcomes

$$\sum_{t \in 1:T} g(\mathcal{Q}_{\leq t}) \;\leq\; T g(\mathcal{Q}_{\leq T}) \;\leq\; T \sum_{t \in 1:T} g(\mathcal{Q}_t),$$

which implies that freezing is no more than a factor T slower than the forward algorithm.

4.A.2.2 Decaying Past

For the decaying past scheme the relative mixing weights of any two past posteriors change from β_t to β_{t+1}, which prevents us from summing them as for uniform past. Implementing decaying past therefore slows down both the evolution of past posteriors and computation of λ_t by a factor of $O(t)$, and increases the required space by the same factor. Fortunately, however, the decaying past scheme can be approximated using a logarithmic number of uniform blocks, as described in Appendix C of [19]. This reduces the slowdown factor from $O(t)$ to $O(\log t)$.[1] Thus, both for sleeping and for freezing, approximated decaying past is only a factor $O(\log T)$ slower than uniform past on T outcomes, and requires only a factor $O(\log T)$ more space.

4.B Loss Bounds

We identify λ_t with the EHMM on $\langle Q_i, E_i, X_i\rangle_{i\geq t}$ with initial distribution λ_t, and with the transition and production functions of $\mathcal{A}$. So in particular $\lambda_1 = \mathcal{A}$. For convenience, we shorten $(\lambda_t)^{\mathsf{fr}}_{\mathcal{C}}(x_{\mathcal{C}})$ to $\lambda^{\mathsf{fr}}_t(x_{\mathcal{C}})$ and $(\lambda_t)^{\mathsf{sl}}_{\mathcal{C}}(x_{\mathcal{C}})$ to $\lambda^{\mathsf{sl}}_t(x_{\mathcal{C}})$. Thus, among others, $\lambda_t(x_t) = \lambda^{\mathsf{sl}}_t(x_t) = \lambda^{\mathsf{fr}}_t(x_t)$.

4.B.1. Lemma. *For any $\mathcal{C} \subseteq t{:}T$, interpreting $\lambda_0(\cdot|x_0)$ as λ_1,*

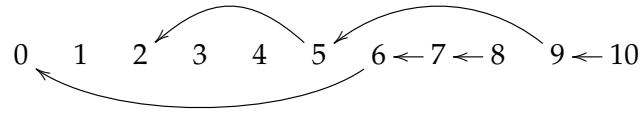
$$\lambda^{\mathsf{f/s}}_t(x_{\mathcal{C}}) = \sum_{j\in 0:t-1} \beta_t(j)\lambda^{\mathsf{f/s}}_j(x_{\mathcal{C}}|x_j).$$

Proof. Let π^t_j denote the past posterior π_j at the beginning of round t. Thus for freezing $\pi^t_j = \pi_j$, and for sleeping π^t_j is π_j evolved $t-j$ steps. Then by definition $\lambda_t(x_{\mathcal{C}}) = \sum_{j=0}^{t-1} \beta_t(j)\pi^t_{j+1}(x_{\mathcal{C}})$. The operations $(\cdot)^{\mathsf{fr}}$ and $(\cdot)^{\mathsf{sl}}$ distribute over taking mixtures. The lemma follows from the fact that $(\pi^t_j)^{\mathsf{sl}}(x_{\mathcal{C}}) = \pi^{\mathsf{sl}}_j(x_{\mathcal{C}})$ and $(\pi^t_j)^{\mathsf{fr}}(x_{\mathcal{C}}) = \pi^{\mathsf{fr}}_j(x_{\mathcal{C}})$. □

Proof of Theorem 4.5.2. For any t, we view the mixing scheme β_t as defining the distribution of a randomised choice $j_t \in 0{:}(t-1)$ for the predecessor of the tth outcome. Let $j_{>k} := j_{k+1:T} = (j_{k+1}, \ldots, j_T)$ denote a

[1] In [19] it is suggested to weight each block of posteriors $\pi_{[j_1,j_2-1]}$ by $(j_2 - j_1)\beta_t(j_1)$. It seems that a marginal improvement is possible by weighting by $\sum_{j_1\leq j<j_2}\beta_t(j)$ instead, which can be implemented equally efficiently for decaying past.

Figure 4.7 Notation example. $T = 10$, $k = 4$, $j_{>k} = (2,0,6,7,5,9)$, $S(j_{>k}) = \{6\}$, $R_2(j_{>k}) = \{2,5,9,10\}$.



vector of the choices beyond turn k. Unfortunately, some choices of $j_{>k}$ are inconsistent with any partition, because an element can only have one successor in a partition. Thus $j_{>k}$ is inconsistent with any partition if $j_m = j_n > 0$ for $k < m \neq n \leq T$. Let the predicate $I(j_{>k})$ be true iff $j_{>k}$ is consistent with some partition.

Some elements of $j_{>k}$ may indicate the start of a new cell of the partition. Let $S(j_{>k})$ denote the set of times when $j_{>k}$ prescribes to start a new cell, i.e. $S(j_{>k}) \coloneqq \{t \in k+1{:}T \mid j_t = 0\}$. For an example, consult Figure 4.7.

Consistent values of $j_{>k}$ specify the last part of a partition. For any $1 \leq t \leq k$, we may ask which of the times $k+1{:}T$ will be put in the same cell as t. Let $R_t(j_{>k})$ denote this set, including t. For convenience, we abbreviate

$$\begin{aligned} \beta(j_{>k}) &\coloneqq \prod_{t \in k+1:T} \beta_t(j_t), \\ W(j_{>k}) &\coloneqq \prod_{i \in S(j_{>k})} \lambda_1^{\mathsf{f/s}}(x_{R_i(j_{>k})}), && \text{and} \\ U_l(j_{>k}) &\coloneqq \prod_{i \in 1:l} \lambda_i^{\mathsf{f/s}}(x_{R_i(j_{>k})}) && \text{for all } l \leq k, \end{aligned}$$

to name the intermediate debris arising from the incremental reduction of $\mathrm{P}_{\mathcal{A}}^{\mathsf{f/s}}(x_{1:T})$. W-terms deal with cells that are completely specified by $j_{>k}$, while U-terms keep track of the remaining partially specified cells. The proof proceeds by downward induction on k, with induction hypothesis

$$\prod_{i \in 1:T} \lambda_i(x_i) \geq \sum_{j_{>k}:I(j_{>k})} \beta(j_{>k}) W(j_{>k}) U_k(j_{>k}). \tag{4.10}$$

For the base case $k = T$ the hypothesis holds with equality, and for $k = 0$ the hypothesis is equivalent to the desired result (4.5). It remains

to verify that it holds for $k-1 \geq 0$ if it holds for k. To this end, fix $k \geq 1$. To prove (4.10), it suffices to show that for consistent $j_{>k}$

$$W(j_{>k})U_k(j_{>k}) \geq \sum_{j_k : I(j_{\geq k})} \beta_k(j_k) W(j_{\geq k}) U_{k-1}(j_{\geq k}),$$

where $j_{\geq k}$ denotes $j_{k:T}$, i.e. j_k followed by $j_{>k}$. We expand the last factor of $U_k(j_{>k})$ using Lemma 4.B.1, and bound

$$\begin{aligned} U_k(j_{>k}) &= \sum_{j_k \in 0:k-1} \beta_k(j_k) \lambda^{\mathsf{f/s}}_{j_k}(x_{R_k(j_{>k})} | x_{j_k}) U_{k-1}(j_{>k}) \\ &\geq \sum_{j_k : I(j_{\geq k})} \beta_k(j_k) \lambda^{\mathsf{f/s}}_{j_k}(x_{R_k(j_{>k})} | x_{j_k}) U_{k-1}(j_{>k}). \end{aligned}$$

Observe that $R_t(j_{>k}) = R_t(j_{\geq k})$ for all $1 \leq t < k$ except $t = j_k$. There are two cases. If $j_k = 0$, then

$$U_{k-1}(j_{>k}) \;=\; U_{k-1}(j_{\geq k}) \quad \text{and} \quad W(j_{>k}) \lambda^{\mathsf{f/s}}_1(x_{R_k(j_{>k})}) \;=\; W(j_{\geq k}).$$

On the other hand if $j_k > 0$ then $W(j_{>k}) = W(j_{\geq k})$. For consistent $j_{\geq k}$, $U_{k-1}(j_{>k})$ contains the factor $\lambda^{\mathsf{f/s}}_{j_k}(x_{j_k})$, which implies that

$$\lambda^{\mathsf{f/s}}_{j_k}(x_{R_k(j_{>k})} | x_{j_k}) U_{k-1}(j_{>k}) = U_{k-1}(j_{\geq k}). \qquad \square$$

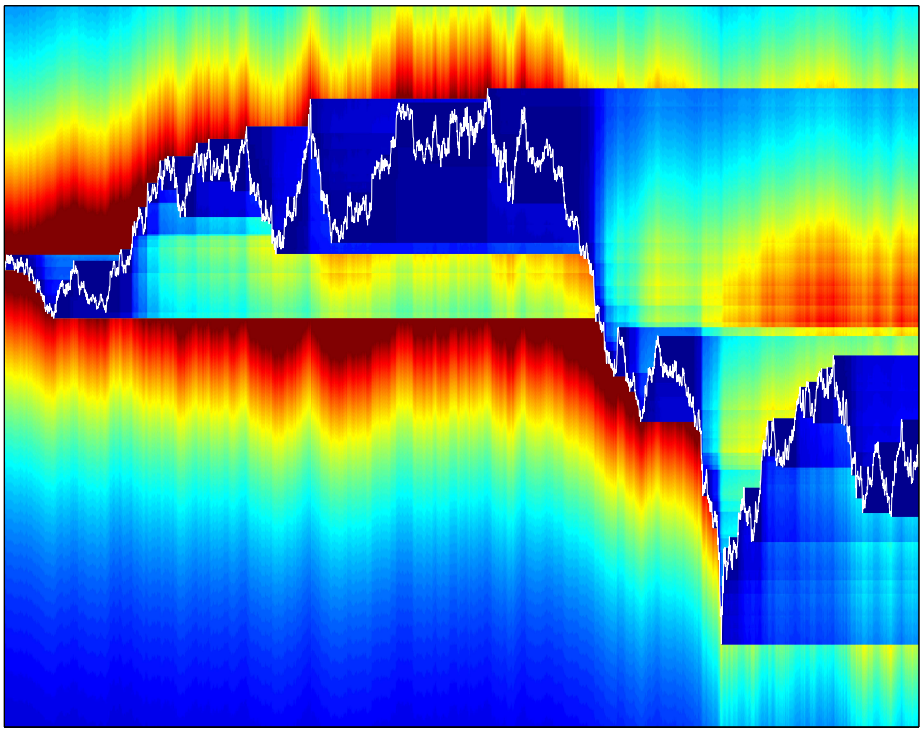
4.C Invariance

Proof of Theorem 4.5.1. Let μ^1 and μ^2 be distributions on $\mathcal{Q}^1$ and $\mathcal{Q}^2$. We overload notation, and write μ^1 and μ^2 for the EHMMs $\mathcal{A}^1$ and $\mathcal{A}^2$ with initial distribution replaced by μ^1 and μ^2. Recall that μ^1 and μ^2 are equivalent if $\mu^1(e_{1:T}) = \mu^2(e_{1:T})$ for all $e_{1:T}$. Thus, $\mathcal{A}^1$ and $\mathcal{A}^2$ are equivalent iff $\mathrm{p}^1_\circ$ and $\mathrm{p}^2_\circ$ are equivalent.

To prove the theorem, we need to prove that equivalence is preserved by all the operations that EPP performs, i.e. taking mixtures, performing loss update and performing state evolution. Mixtures of equivalent distributions are equivalent, since mixing and marginalisation commute. For loss update, note that $\mathrm{p}^{e_1}_1(x_1) = \mu^1(x_1 | e_{1:T}) = \mu^2(x_1 | e_{1:T})$ for all $\mathrm{p}^{\mathcal{E}}_1$ and all $e_{1:T}$. Finally, for state evolution, the claim follows from $(\mathrm{p}_{\rightarrow} \circ \mu)(e_{1:T}) = \mu(E_{2:T+1} = e_{1:T})$. $\square$

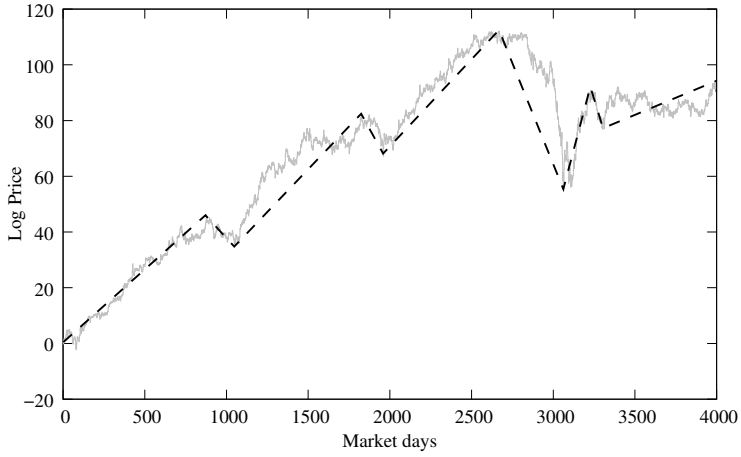
Chapter 5

Switching Investments



Abstract We present a simple online two-way trading algorithm that exploits fluctuations in the unit price of an asset. Rather than analysing worst-case performance under some assumptions, we prove a novel, unconditional performance bound that is parameterised either by the actual dynamics of the price of the asset, or by a simplifying model thereof. The algorithm processes T prices in $O(T^2)$ time and $O(T)$ space, but if the employed prior density is exponential, the time requirement reduces to $O(T)$. The result translates to the prediction with expert advice framework, and has applications in data compression and hypothesis testing.

Figure 5.1 An example play Λ for Nature (solid gray), with a regularised trend line Λ' (dashed black).



5.1 Introduction

We consider a two-player game played between *Investor* and *Nature*. Investor starts out with one unit of cash. At each time, Investor decides which fraction of his current capital to invest in an asset (denoted A), and how much to keep in his boot (denoted B). Nature, on the other hand, chooses the price of the asset.

A play for Nature is a function $\Lambda : [0, T] \to \mathbb{R}$ that specifies the natural logarithm of the unit price of A as a function of time. The end-time T is part of Nature's move and unknown to Investor. An example play is shown in Figure 5.1.

Investor's *payoff* is defined as the natural logarithm of his capital at the end-time T, where shares owned are valued at the final logprice $\Lambda(T)$. In hindsight, it would have been optimal for Investor to follow the strategy S_Λ that invests all capital in A at local minima of Λ, and liquidates all shares into B at local maxima. Let $\boldsymbol{z} = z_0, \ldots, z_m$ denote the sequence of logprices at local extrema of Λ, with $z_0 = \Lambda(0)$ and $z_m = \Lambda(T)$. The payoff of the strategy S_Λ thus equals

$$S_\Lambda * \Lambda \;:=\; \sum_{1 \le i \le m} \max\{0, z_i - z_{i-1}\}.$$

We construct a foresight-free, computationally efficient strategy π that

guarantees payoff $\pi{*}\Lambda$ close to $S_\Lambda{*}\Lambda$. The definition of π relies on the selection of a probability density function on $[0,\infty)$ that for convenience we identify with π itself (see Section 5.2), and we abbreviate $-\ln\pi(h)$ to $\ell(h)$. We then prove

$$\pi{*}\Lambda \;\geq\; S_\Lambda{*}\Lambda - \sum_{1\leq i\leq m} \ell\big(|z_i - z_{i-1}|\big) - (m-1)c_\pi - \ln 2 - 2\epsilon_\pi, \quad (5.1)$$

where c_π and ϵ_π are two constants that depend on π. Thus the payoff of π on Λ falls short of the optimum by an overhead that depends on the complexity of Λ, measured in terms of both the length of the vector z, and the sizes of its entries. The bound is entirely independent of the time scale T.

When Λ is simple, i.e. has few large fluctuations, (5.1) shows that π exploits almost all achievable payoff. The bound degenerates when Λ sports many small fluctuations, for which the overhead $\ell(x)$ exceeds the benefit x of trading. However, we prove that for any *regularisation* Λ' of Λ, as illustrated by the dashed line in Figure 5.1 and defined precisely in Section 5.3.2, π's payoff satisfies

$$\pi{*}\Lambda' \;\leq\; \pi{*}\Lambda. \quad (5.2)$$

Thus, we may pretend that Nature actually played Λ', and apply the bound (5.1) with Λ' in place of Λ. In fact the regulariser Λ' may be interpreted as a *model* for Nature's play Λ. The most complex model then yields the bound as presented in (5.1), but we may now concern other models, that strike a better balance between model complexity and goodness of fit. Such tradeoff models will usually yield better bounds. In conclusion, if in hindsight a simple regulariser can be found with large payoff, then π will collect most of that payoff as well.

5.1.1. Example. Let Λ and Λ' be, respectively, the play for Nature and the regulariser shown in Figure 5.1. The extrema of the regulariser are given by $z' = (0, 42, 36, 82, 68, 112, 57, 90, 77, 90)$. Then $S_{\Lambda'}{*}\Lambda' = (42-0) + (82-36) + \ldots + (90-77) = 178$. Now we select the exponential density listed in Table 5.1 for the definition of π; the values for c_π and ϵ_π are also listed there. We can now apply bounds (5.2) and (5.1) to find

$$\pi{*}\Lambda \;\geq\; \pi{*}\Lambda' \;\geq\; 178 - 64.8 - 8\cdot 0.034 - \ln 2 - 2\cdot 3.40 \;\approx\; 105.4.$$

Note that there may be choices of Λ' for which the bound is better, and even for the optimal choice of Λ' the strategy π may perform substantially better than our bound indicates. The actual payoff of π on these data is $\pi*\Lambda = 175.4$. $\diamond$

5.1.0.3 Applications and Related Work.

Our model and its analysis are phrased in financial terms. However, it applies much more widely. We list four examples.

One-Way Trading and Two-Way Trading. This is the most direct example. We let Λ be the logarithm of the exchange rate between any two assets, say dollar and yen. If we forbid selling A, we obtain the setting called *One-way Trading*. Efficient algorithms with minimax payoff for one-way trading under various restrictions on Nature's play Λ are known. E.g. fixed daily price growth range [29], fixed price range [53] and bounded quadratic variation [43]. Two-way trading guarantees are derived in [38] by iterating a unidirectional trading algorithm back and forth. Both the algorithms and the bounds are parametrised by the restrictions placed on Nature's play.

Our results are of a different kind. First, no restrictions are placed on Nature's play. Second, our guarantees are expressed in terms of Nature's actual play (or a regularisation thereof), and hence remain informative when Nature does not play to ruin Investor.

Prediction with Expert Advice. Two experts, say A and B sequentially issue predictions. We denote their cumulative loss at time t by $L_A(t)$ and $L_B(t)$. We let $\Lambda(t) = L_B(t) - L_A(t)$. In prediction tasks with so-called *mixable loss* [182], guarantees for our financial game directly translate to expert performance bounds and vice versa. Efficient strategies include the seminal *Fixed Share* [80], the *Switching Method* [180], and the *Switch Distribution* and its derivatives in Chapter 3 and [178]. These algorithms guarantee payoff $\rho*\Lambda \geq S_{\Lambda'}*\Lambda' - O(m' \ln T)$ for each Λ' with m' blocks. The logarithmic dependence of the bound on the time T of these algorithms means that for any arbitrary number h, if by switching just a single time the payoff could be improved by h, there is a sample size T such that these algorithms are not able to exploit this.

Variable Share [80] switches based on the losses L^{A} and L^{B}. Its payoff guarantee depends logarithmically on the loss of the best reference strategy with m' blocks. However, its analysis assumes so-called *bounded loss*, and does not apply to financial games (which involve logarithmic loss, which is unbounded).

Prefix Coding/Compression. Fix two prefix codes A and B for a sequence of outcomes $x_1, \ldots, x_T$. Let $L_{\mathrm{A}}(t)$ and $L_{\mathrm{B}}(t)$ denote the code-length of A and B on the outcomes $x_1, \ldots, x_t$ measured in nats. Now let $\Lambda(t) = L_{\mathrm{B}}(t) - L_{\mathrm{A}}(t)$. It is well-known that we can build a prefix code that attains code length $\ln(2) + \min\{L_{\mathrm{A}}(T), L_{\mathrm{B}}(T)\}$ on the data. When different codes are good for different segments of the data, we observe fluctuation in Λ. Using standard information-theoretic methods, e.g. [34], our financial prediction scheme can be transformed into a prefix code that exploits these fluctuations.

Hypothesis Testing. We are given a *null hypothesis* P_0 and an *alternative hypothesis* P_1. Both candidate hypotheses are probabilistic models for some sequence of observations $x_1, x_2, \ldots, x_T$. Let

$$\Lambda(t) \;=\; \ln \frac{P_1(x_1, \ldots, x_t)}{P_0(x_1, \ldots, x_t)}$$

be the loglikelihood ratio between P_1 and P_0. Thus Λ measures the amount of evidence against the null hypothesis and can be used as a test statistic. Traditionally [15], we choose a threshold $\tau > 0$ and reject the null hypothesis when $\Lambda(T) \geq \tau$, an event that is extremely unlikely under P_0. The case where $\Lambda(T)$ is below the threshold τ, while $\Lambda(t) \geq \tau$ at some earlier time t is considered in [162, 40], and tests are presented that lose as little evidence as possible while remaining unbiased. These tests are based on strategies that switch only once, and resemble strategies for one-way trading. By the same method, our strategy induces a fair test statistic that can be used to reject P_0 whenever Λ fluctuates heavily; an event that is also unlikely under P_0.

5.1.0.4 Outline.

We explicate the setting and describe the strategy π for Investor in Section 5.2. We analyse the payoff of π and prove our payoff guarantee in

Section 5.3. We then show how to implement the strategy π efficiently in Section 5.4.

5.2 Setting

We introduce the details of our financial game. We first review Nature's play Λ. We then construct strategies for Investor, culminating in the definition of the strategy π. We conclude this section with a lemma that simplifies all later proofs by exploiting the symmetry between A and B.

5.2.1 Nature's Play Λ

A play for Nature is a logprice function $\Lambda : [0, T] \to \mathbb{R}$. The end-time T is part of Nature's move, and unknown to Investor.

For simplicity, we restrict attention to the setting where Λ is discrete, i.e. piecewise constant with jumps at integer times. This is sufficient for the practical scenario where Λ is monitored intermittently (albeit possibly very often). Later in the analysis it will be convenient for technical reasons to generalise to piecewise continuous plays for Nature with finitely many local extrema and finitely many discontinuities; nevertheless ultimately we remain concerned with the discrete setting only.

Our results do extend quite readily to the wide class of càdlàg logprice functions (right-continuous with left limits). These encompass continuous time models that are often considered in the financial literature, such as Brownian motion with drift, etc. Such theoretically interesting generalisations are deferred to future publications.

5.2.2 Investor's Strategy π

We now construct the strategy π for Investor in three stages. Two basic strategies exist. Strategy A invests the initial unit capital in the asset, whereas strategy B keeps all capital in the boot. At the end of the game, all shares are valued at the final logprice $\Lambda(T)$. The payoffs, defined as Investor's final logcapital, of the basic strategies equal

$$\mathrm{A}{*}\Lambda \;:=\; \Lambda(T) - \Lambda(0) \qquad \text{and} \qquad \mathrm{B}{*}\Lambda \;:=\; 0.$$

Since we use logprice differences extensively, we abbreviate $\Lambda(t) - \Lambda(s)$ to $\Lambda|_s^t$.

5.2.2.1 Time-switched Strategies.

From these basic strategies A and B we construct more interesting strategies. Let $\boldsymbol{t} = t_0, t_1, t_2, t_3, \ldots$ be a sequence of times such that $0 = t_0 \leq t_1 \leq t_2 \leq \ldots$ The strategy $\boldsymbol{t}^{\mathrm{A}}$ switches at times $\boldsymbol{t}$ starting with A. That is, $\boldsymbol{t}^{\mathrm{A}}$ invests all capital in A until time t_1. At that time it sells all shares, and keeps all money in B until time t_2. Then it again invests all capital in A until time t_3 etc. Symmetrically, $\boldsymbol{t}^{\mathrm{B}}$ is the strategy that switches at times $\boldsymbol{t}$ starting with B. Thus the payoffs of $\boldsymbol{t}^{\mathrm{A}}$ and $\boldsymbol{t}^{\mathrm{B}}$ when Nature plays Λ are

$$\boldsymbol{t}^{\mathrm{A}}{*}\Lambda \;:=\; \sum_{i=0}^{\infty} \Lambda|_{T\wedge t_{2i}}^{T\wedge t_{2i+1}} \qquad \text{and} \qquad \boldsymbol{t}^{\mathrm{B}}{*}\Lambda \;:=\; \sum_{i=0}^{\infty} \Lambda|_{T\wedge t_{2i+1}}^{T\wedge t_{2i+2}}.$$

Of course, a good time switch sequence $\boldsymbol{t}$ for Investor depends on Nature's unknown move Λ. However, Investor may hedge by dividing his initial capital according to some prior distribution ρ on the switch time sequence $\boldsymbol{t}$, and construct time-switched strategies ρ^{A} and ρ^{B} with payoffs

$$\begin{aligned} \rho^{\mathrm{A}}{*}\Lambda \;&:=\; \ln \int \exp\bigl(\boldsymbol{t}^{\mathrm{A}}{*}\Lambda\bigr)\, \mathrm{d}\rho(\boldsymbol{t}) \qquad \text{and} \\ \rho^{\mathrm{B}}{*}\Lambda \;&:=\; \ln \int \exp\bigl(\boldsymbol{t}^{\mathrm{B}}{*}\Lambda\bigr)\, \mathrm{d}\rho(\boldsymbol{t}), \end{aligned}$$

and the meta strategy ρ with payoff

$$\rho{*}\Lambda \;:=\; \ln\Bigl(\tfrac{1}{2}\exp\bigl(\rho^{\mathrm{A}}{*}\Lambda\bigr) + \tfrac{1}{2}\exp\bigl(\rho^{\mathrm{B}}{*}\Lambda\bigr)\Bigr).$$

5.2.2.2 Price-switched Strategies.

Price-switched strategies decide when to trade based on the logprice $\Lambda(t)$ instead of the time t itself. This renders their payoff independent of the time-scale. Fix a sequence of nonnegative reals $\boldsymbol{\delta} = \delta_1, \delta_2, \ldots$ We denote by $\boldsymbol{\delta}^{\mathrm{A}}$ the strategy that initially invests all capital in A, and waits until the first time s_1 where the logprice difference $\Lambda|_0^{s_1}$ is at least δ_1. It then sells all shares and puts the money into B, until the first subsequent

time s_2 that the logprice difference $\Lambda|_{s_1}^{s_2}$ is at most $-\delta_2$. Then it invests all capital into A again, until the logprice difference $\Lambda|_{s_2}^{s_3}$ is at least δ_3, etc. The strategy $\boldsymbol{\delta}^{\mathrm{B}}$ is defined symmetrically, with switching times $r_0, r_1, \ldots$ The switching time sequences $\boldsymbol{s}$ and $\boldsymbol{r}$ are obtained as follows. First $s_0 = r_0 = 0$. Then recursively, for even i

$$s_i := \min\{t \geq s_{i-1} \mid \Lambda|_{s_{i-1}}^{t} \geq +\delta_i\} \quad r_i := \min\{t \geq r_{i-1} \mid \Lambda|_{r_{i-1}}^{t} \leq -\delta_i\}$$

and for odd i

$$s_i := \min\{t \geq s_{i-1} \mid \Lambda|_{s_{i-1}}^{t} \leq -\delta_i\} \quad r_i := \min\{t \geq r_{i-1} \mid \Lambda|_{r_{i-1}}^{t} \geq +\delta_i\}.$$

Both $\boldsymbol{s}$ and $\boldsymbol{r}$ are a function of $\boldsymbol{\delta}$ and Λ and satisfy $\boldsymbol{s}(\boldsymbol{\delta}, \Lambda) = \boldsymbol{r}(\boldsymbol{\delta}, -\Lambda)$. By convention, the minimum is infinite if no suitable successor time exists in the domain of Λ, i.e before time T. The payoffs of $\boldsymbol{\delta}^{\mathrm{A}}$ and $\boldsymbol{\delta}^{\mathrm{B}}$ are given by

$$\boldsymbol{\delta}^{\mathrm{A}} * \Lambda \;:=\; \boldsymbol{s}^{\mathrm{A}} * \Lambda \qquad \text{and} \qquad \boldsymbol{\delta}^{\mathrm{B}} * \Lambda \;:=\; \boldsymbol{r}^{\mathrm{B}} * \Lambda.$$

The strategy $\boldsymbol{\delta}^{\mathrm{A}}$ has the following property. Whenever it sells its shares, say at time s_i for some odd i, the asset price, and hence its capital, has multiplied by *at least* $\exp(\delta_i) \geq 1$ since the acquisition at time s_{i-1}. This holds irrespective of Nature's play. In particular, between time s_i and s_{i+1} for odd i, the logarithm of its capital equals

$$\Lambda|_{s_0}^{s_1} + \Lambda|_{s_2}^{s_3} + \Lambda|_{s_4}^{s_5} + \ldots + \Lambda|_{s_{i-1}}^{s_i} \;\geq\; \delta_1 + \delta_3 + \delta_5 + \ldots + \delta_i.$$

For each logprice difference sequence $\boldsymbol{\delta}$, the number of switches that is executed, and hence the quality of $\boldsymbol{\delta}^{\mathrm{A}}$ depends on Nature's move Λ. Let $\mathcal{D} = \{\boldsymbol{\delta}^{\mathrm{A}}, \boldsymbol{\delta}^{\mathrm{B}} \mid \boldsymbol{\delta} \in [0, \infty)^{\infty}\}$ be the set of price-switched strategies for Investor.

5.2.2.3 The Strategy π.

Again, we may hedge by dividing our initial capital according to some prior π on $\boldsymbol{\delta}$, and obtain strategies π^{A} and π^{B} with payoffs

$$\pi^{\mathrm{A}} * \Lambda \;:=\; \ln \int \exp\bigl(\boldsymbol{\delta}^{\mathrm{A}} * \Lambda\bigr)\, \mathrm{d}\pi(\boldsymbol{\delta}) \qquad \text{and}$$
$$\pi^{\mathrm{B}} * \Lambda \;:=\; \ln \int \exp\bigl(\boldsymbol{\delta}^{\mathrm{B}} * \Lambda\bigr)\, \mathrm{d}\pi(\boldsymbol{\delta}),$$

and the meta strategy π with payoff

$$\pi{*}\Lambda \;:=\; \ln\Big(\tfrac{1}{2}\exp\big(\pi^{\mathrm{A}}{*}\Lambda\big) + \tfrac{1}{2}\exp\big(\pi^{\mathrm{B}}{*}\Lambda\big)\Big)\,.$$

Note that the price-switched strategies in $\mathcal{D}$ are independent of the time scale, and so are these strategies based on them.

Requirements on π. The above construction works for any prior π. In this chapter we analyse the behaviour of strategies π that satisfy these requirements:

1. π is the independent infinite product distribution of some probability density function on $[0,\infty)$. Since the distinction is always clear, we also denote the univariate density by π.
2. the function $x \mapsto \mathrm{e}^x\pi(x)$ is increasing.
3. the density π is log-convex.

The first requirement ensures that we can hedge capital according to π. The second requirement ensures that paying $-\ln\pi(x)$ to gain x is a better deal when x is larger. The third requirement ensures that we rather pay $-\ln\pi(x+y)$ than $-\ln\pi(x)-\ln\pi(y)$ to gain $x+y$. We use the following consequences in our bounds.

5.2.1. Lemma. *Let π satisfy the requirements 1–3 above. Then*

1. *π is strictly positive.*
2. *π is strictly decreasing.*
3. *$\int_h^\infty \pi(x)\,\mathrm{d}x \geq \pi(h)$ for each $h \geq 0$.*

Proof. Since π is a convex probability density, it is decreasing and thus $0 < \pi(0) = \mathrm{e}^0\pi(0)$. Since $\mathrm{e}^x\pi(x)$ increases, we have $\pi(x) > 0$ for all x. Then, since π is a non-zero convex probability density, it must be *strictly* decreasing. Finally, for $0 \leq h \leq x$ we have $\pi(x) = \pi(x)\mathrm{e}^x\mathrm{e}^{-x} \geq \pi(h)\mathrm{e}^h\mathrm{e}^{-x}$. Therefore $\int_h^\infty \pi(x)\,\mathrm{d}x \;\geq\; \pi(h)\int_h^\infty \mathrm{e}^{h-x}\,\mathrm{d}x \;=\; \pi(h)\,.$ □

The last fact implies that the density $\pi(x) \leq 1$ for all x. Throughout this chapter, we abbreviate $-\ln\pi(x)$ to $\ell(x)$. Thus ℓ is nonnegative, concave and increasing.

Table 5.1 Example priors.

	Fat tail	Pareto	Exponential
$\pi(x)$	$\dfrac{\log(o)}{(x+o)(\log(x+o))^2}$	$(c-1)o^{c-1}(x+o)^{-c}$	$\alpha \mathrm{e}^{-\alpha x}$
Condition	$2 \leq (o-1)\log o$ (Sufficient: $o \geq 2.89$)	$1 < c \leq o$	$0 < \alpha \leq 1$
Parameters	$o = 3$	$c = 2$, $o = 3$	$\alpha = 1/3$
ϵ_π	4.10396	3.55884	3.39788
c_π	0.016645	0.0288849	0.034016

5.2.2. Example. The densities shown in Table 5.1, ordered from heavy to light tails, satisfy all the requirements. ◇

5.2.3 Exploiting Symmetry

Payoff is measured as (the natural logarithm of) Investor's final amount of cash. Of course, cash and asset are intrinsically symmetric. We make this precise as follows. We say that the following pairs of strategies are *dual*

$$\mathrm{A}, \mathrm{B} \qquad \boldsymbol{t}^{\mathrm{A}}, \boldsymbol{t}^{\mathrm{B}} \qquad \rho^{\mathrm{A}}, \rho^{\mathrm{B}} \qquad \rho, \rho \qquad \boldsymbol{\delta}^{\mathrm{A}}, \boldsymbol{\delta}^{\mathrm{B}} \qquad \pi^{\mathrm{A}}, \pi^{\mathrm{B}} \qquad \pi, \pi$$

and vice versa in each case. The meta strategies ρ and π are self-dual.

5.2.3. Lemma (Duality). *Let S and S' be dual strategies. Then for each Λ*

$$S{*}\Lambda \;=\; S'{*}(-\Lambda) + \Lambda|_0^T.$$

Proof. The lemma is trivial for the dual pair A and B. We proceed to prove the lemma for the dual strategies $\boldsymbol{t}^{\mathrm{A}}$ and $\boldsymbol{t}^{\mathrm{B}}$, the other cases follow simply by definition. Recall that $\exp(\Lambda)$ is the asset price in cash per share, so that $\exp(-\Lambda)$ is the price in shares per cash. Thus $\boldsymbol{t}^{\mathrm{B}}{*}(-\Lambda)$ is the log-number of shares resulting from investing one share according to the strategy $\boldsymbol{t}^{\mathrm{A}}$. Finally, $\Lambda|_0^T = \Lambda(T) - \Lambda(0)$ is the result of exchanging cash to asset initially, and asset to cash at the end. □

5.3 Payoff Bound

In this section we prove the payoff guarantees for the strategy π that were given in the introduction. We build towards the statement and proof of a more precise version of the bounds in the following subsections. First, in Section 5.3.1 we show that Nature's worst-case logprice functions are continuous. Then, in Section 5.3.2 we show that Investor's payoff decreases when Nature plays more regular. In Section 5.3.3 we analyse Investor's payoff under a regularity assumption on Λ called γ-separation. Finally, in Section 5.3.4 we show how to establish γ-separation if it does not obtain and establish the bound in the form of Theorem 5.3.8.

5.3.1 Nature Plays a Continuous Logprice Function Λ

We now prove that it is sub-optimal for Nature to play a discontinuous Λ. To do so, we show that Investor's payoff is reduced when Nature eliminates a jump by inserting a linear interpolation. Let Λ have a discontinuity at t. We define Λ', *the t-ironing of* Λ, by $\Lambda'(s) := \Lambda(s)$ for $s < t$, $\Lambda'(s+1) := \Lambda(s)$ for $s > t$, and $\Lambda'(s) := (1+t-s)\Lambda(t-) + (s-t)\Lambda(t)$ for $t \leq s \leq t+1$, where $\Lambda(t-) := \lim_{s \uparrow t} \Lambda(s)$. This definition is illustrated by Figure 5.2.

5.3.1. Theorem (Continuous Free Lunch). *Fix any play for Nature Λ with a discontinuity at time t, and let Λ' be the t-ironing of Λ. Then*

$$\pi * \Lambda' \;\leq\; \pi * \Lambda.$$

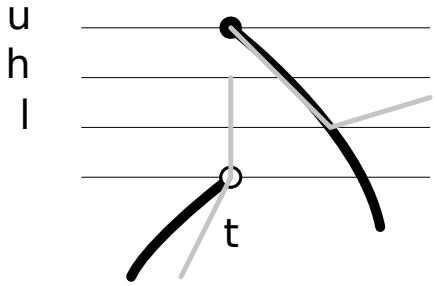
Proof. See Figure 5.2. By duality (Lemma 5.2.3), we may assume that the jump is upward. Obviously, any strategy δ' that does not switch at time t on Λ has identical payoff on Λ and Λ'. Now consider any strategy $\delta' = (\ldots, h, h-l, \ldots)$, where h prompts a switch at time t on Λ. We now modify the strategy to $\delta = (\ldots, h, u-l, \ldots)$ and we compare the term corresponding to δ' in the integral for $\pi * \Lambda'$ to the term corresponding to δ in the integral for $\pi * \Lambda$:

$$\frac{\exp(\delta' * \Lambda')\pi(\delta')}{\exp(\delta * \Lambda)\pi(\delta)} \;=\; \frac{\exp(h-l)\pi(h-l)}{\exp(u-l)\pi(u-l)} \;\leq\; 1,$$

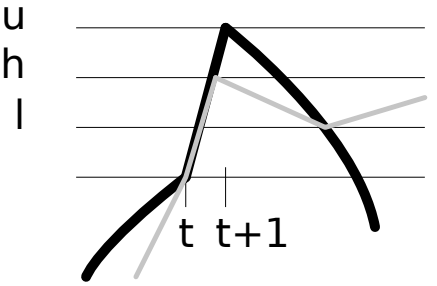
where the inequality uses that $\mathrm{e}^h \pi(h)$ is increasing (see Section 5.2.2.3). The proof follows by observing that the mapping that takes δ' to δ is a translation. □

Figure 5.2 Worst-case plays for Nature are continuous.

(a) Nature's move Λ with jump at time t, and strategy $\delta = (\ldots, h, u-l, \ldots)$.



(b) The t-ironing Λ' of Λ, and strategy $\delta' = (\ldots, h, h-l, \ldots)$



When Investor follows the strategy π, there is no benefit for Nature to playing a logprice function Λ with jumps. Without loss of generality we henceforth restrict Nature to continuous plays. This simplifies analysis considerably, as it allows us to assume that switches specified by any δ occur at *exactly* the specified logprices.

5.3.2 Ordering by Regularity

Given a move for Nature $\Lambda : [0, T] \to \mathbb{R}$, we say that another move $\Lambda' : [0, T'] \to \mathbb{R}$ is *more regular* than Λ, denoted $\Lambda' \preccurlyeq \Lambda$, if there is a monotonic function $f : [0, T'] \to [0, T]$ such that $f(0) = 0$, $f(T') = T$ and $\Lambda' = \Lambda \circ f$. That is, the price levels of the *regularisation* Λ' are a subsequence of the price levels of Nature's move Λ, with the same initial and final price, but potentially less fluctuation. We now show that by following a fixed price-switched strategy, Investor gets richer whenever Nature's move is less regular.

5.3.2. Theorem (Monotonicity). *For each price-switched strategy $S \in \mathcal{D}$ and continuous logprice functions Λ and Λ'*

$$\Lambda' \preccurlyeq \Lambda \qquad \textit{implies} \qquad S{*}\Lambda' \leq S{*}\Lambda.$$

Proof. First note that $\Lambda' \preccurlyeq \Lambda$ iff $-\Lambda' \preccurlyeq -\Lambda$. So that by Lemma 5.2.3 it suffices to prove the theorem for the strategies in $\mathcal{D}$ that start with A. We proceed by induction on the number of switches executed by the strategy δ^{A} on the regulariser Λ'. For the base case, suppose this

number is zero, i.e $\Lambda'|_0^t < \delta_1$ for each $0 \le t \le T'$. Let $m \ge 1$ denote the number of blocks of $\boldsymbol{\delta}^{\mathrm{A}}$ on Λ. There are two cases. If m is even then $\boldsymbol{\delta}^{\mathrm{A}}$ follows B on the last block. Since $\delta_1 > \Lambda'|_0^{T'}$

$$\boldsymbol{\delta}^{\mathrm{A}} * \Lambda \;=\; \sum_{1 \le i < m \text{ odd}} \delta_i \;\ge\; \delta_1 \;>\; \Lambda'|_0^{T'} \;=\; \boldsymbol{\delta}^{\mathrm{A}} * \Lambda'.$$

If m is odd, then $\boldsymbol{\delta}^{\mathrm{A}}$ follows A on the last block. Again invoking Lemma 5.2.3, we get

$$\boldsymbol{\delta}^{\mathrm{A}} * \Lambda \;=\; \sum_{1 \le i < m \text{ even}} \delta_i + \Lambda|_0^T \;\ge\; \Lambda|_0^T \;=\; \Lambda'|_0^{T'} \;=\; \boldsymbol{\delta}^{\mathrm{A}} * \Lambda'.$$

To prove the induction step, suppose a switch is executed, i.e. the first difference δ_1 is present in the regulariser Λ', and hence also in Nature's play Λ, then the strategy $\boldsymbol{\delta}^{\mathrm{A}}$ switches at price level $\Lambda(0) + \delta_1$ on either play, resulting in the same capital. The switches may occur at different times on Λ and Λ'. Nevertheless, the induction hypothesis applies to the tails of the plays since the remainder of the regulariser Λ' is more regular than the remainder of Nature's move Λ. □

Since the theorem holds pointwise in $\mathcal{D}$, it also holds for the mixture strategy π.

5.3.3 With γ-Separation

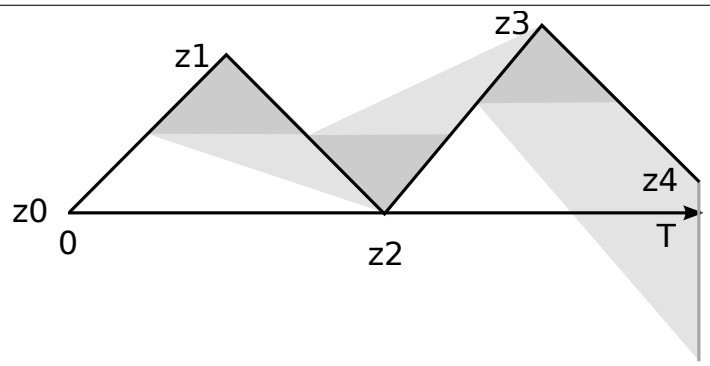
Fix a logprice function Λ. Throughout this section, we use the following notation:

5.3.3. Definition. We denote by $\boldsymbol{z} = z_0, z_1, \ldots, z_m$ the sequence of logprices at the local extrema of Λ (attained or not), with $z_0 = \Lambda(0)$ and $z_m = \Lambda(T)$, and we say that Λ has *m blocks*. Let $\boldsymbol{\Delta} = \Delta_1, \ldots, \Delta_m$ denote the sequence of absolute logprice differences, i.e. $\Delta_i := |z_i - z_{i-1}|$.

5.3.4. Definition. We say that Λ has *γ-separation* if $\Delta_1, \Delta_m \ge \gamma$ and $\Delta_i \ge 2\gamma$ for each $1 < i < m$. That is, the border optima have logprice difference at least γ with the border and each subsequent pair of local extrema has at least logprice difference 2γ.

We now analyse the payoff of the strategy π, assuming that Λ has γ-separation.

Figure 5.3 Domain of integration example. Some Λ, with $m = 4$, is shown in black. The height of the dark gray triangles equals γ. This Λ has γ-separation. In particular $\Delta_2 = z_1 - z_2 = 2\gamma$. Theorem 5.3.5 integrates over the strategies that are optimal for log-price functions in the light gray region.



5.3.5. Theorem (γ-Separation Payoff). *For each Λ with γ-separation*

$$\pi{*}\Lambda \;\geq\; \underbrace{\sum_{1\leq i\leq m} (z_i - z_{i-1})_+}_{\textit{gain}} \;-\; \underbrace{\sum_{1\leq i\leq m} \ell(\Delta_i)}_{\textit{complexity penalty}} \;+\; (m-1)\;\underbrace{\ln(1-\mathrm{e}^{-\gamma})}_{\textit{overhead per switch}} \;-\; \underbrace{\ln 2}_{\textit{parity}}\,.$$

Proof. We saw in Section 5.2.3 that π is self-dual, so by symmetry (Lemma 5.2.3) we may assume that $z_0 \leq z_1$. As our first bound, we use $\pi{*}\Lambda \geq \pi^{\mathrm{A}}{*}\Lambda - \ln 2$. Recall that the payoff $\pi^{\mathrm{A}}{*}\Lambda$ is defined as $\ln \int \exp\bigl(\boldsymbol{\delta}^{\mathrm{A}}{*}\Lambda\bigr)\,\mathrm{d}\pi(\boldsymbol{\delta})$. As the next step, we re-parameterise the integral by introducing variables $\boldsymbol{h}$, with $h_i := z_0 - \sum_{1\leq j\leq i}(-1)^j\delta_j$. That is, h_i is the logprice at the ith switch of $\boldsymbol{\delta}^{\mathrm{A}}$. Then we obtain a lower bound by restricting the domain of integration. For $1 \leq i < m$ we restrict $h_i \in [z_i - \gamma, z_i]$ for odd i and $h_i \in [z_i, z_i + \gamma]$ for even i. Thus, we keep all prior mass on strategies that switch at logprices h_i that are at most γ nats short of the optimal switching logprice level z_i. We restrict the last logprice to $h_m \in [z_m, \infty)$ for even m and $h_m \in (-\infty, z_m]$ for odd m. This ensures that we do not switch between h_{m-1} and z_m. Thus, we only integrate over those strategies that closely follow Λ, as illustrated by Figure 5.3. We first consider even m. Then

$$\pi^{\mathrm{A}} * \Lambda \;\geq\; \ln \int_{z_1-\gamma}^{z_1} \mathrm{e}^{h_1-h_0}\pi(h_1-h_0) \int_{z_2}^{z_2+\gamma} \pi(h_1-h_2) \int_{z_3-\gamma}^{z_3} \mathrm{e}^{h_3-h_2}\pi(h_3-h_2)\cdots \cdots \int_{z_{m-1}}^{z_{m-1}+\gamma} \pi(h_{m-2}-h_{m-1}) \int_{z_m}^{\infty} \mathrm{e}^{z_m-h_{m-1}}\pi(h_m-h_{m-1})\,\mathrm{d}\boldsymbol{h}$$

Apply the tail probability bound (Lemma 5.2.1(3)) to the innermost integral to get

$$\int_{z_m}^{\infty} \mathrm{e}^{z_m-h_{m-1}}\pi(h_m-h_{m-1})\,\mathrm{d}h_m \;\geq\; \mathrm{e}^{z_m-h_{m-1}}\pi(z_m-h_{m-1}).$$

Since $|h_i - h_{i-1}| \leq \Delta_i$ and π decreases (Lemma 5.2.1(2)) we get

$$\pi^{\mathrm{A}} * \Lambda \;\geq\; \ln \prod_{1\leq i\leq m} \pi(\Delta_i) + \ln\left(\mathrm{e}^{z_m-z_0} \int_{z_1-\gamma}^{z_1} \mathrm{e}^{h_1} \int_{z_2}^{z_2+\gamma} \mathrm{e}^{-h_2} \int_{z_3-\gamma}^{z_3} \mathrm{e}^{h_3} \cdots \int_{z_{m-2}-\gamma}^{z_{m-2}} \mathrm{e}^{h_{m-2}} \int_{z_{m-1}}^{z_{m-1}+\gamma} \mathrm{e}^{-h_{m-1}}\,\mathrm{d}\boldsymbol{h} \right)$$

Now all integrals have become independent. Rewrite odd/even instances like

$$\int_{z_1-\gamma}^{z_1} \mathrm{e}^{h_1}\,\mathrm{d}h_1 \;=\; \mathrm{e}^{z_1}(1-\mathrm{e}^{-\gamma}) \quad\text{and}\quad \int_{z_2}^{z_2+\gamma} \mathrm{e}^{-h_2}\,\mathrm{d}h_2 \;=\; \mathrm{e}^{-z_2}(1-\mathrm{e}^{-\gamma}).$$

By rearranging terms we obtain

$$\pi^{\mathrm{A}} * \Lambda \;\geq\; \sum_{1\leq i\leq m} (z_i - z_{i-1})_+ \;-\; \sum_{1\leq i\leq m} \ell(\Delta_i) \;+\; (m-1)\ln(1-\mathrm{e}^{-\gamma}).$$

The case for odd m is analogous. □

5.3.4 Establishing γ-Separation

Say we have a Λ with γ-separation, and hence a performance guarantee by Theorem 5.3.5. If γ is small, then a better bound can be obtained by

Algorithm 5.1 The ϵ-pruning algorithm

1: $u \leftarrow (2\epsilon - \Delta_1)_+ \cdot \operatorname{sign}(z_1 - z_0)$.
2: $v \leftarrow (2\epsilon - \Delta_m)_+ \cdot \operatorname{sign}(z_m - z_{m-1})$.
3: $\boldsymbol{z} \leftarrow (z_0, z_1 + u, z_2 + u, \ldots, z_m + u + v)$ ▷ *Ensure* $\Delta_1, \Delta_m \geq 2\epsilon$
4: **while** the minimal Δ_i is (strictly) less than 2ϵ **do**
5: $\quad \boldsymbol{z} \leftarrow (z_0, z_1, \ldots, z_{i-2}, z_{i+1}, \ldots, z_m)$ ▷ *See Figure 5.4*
6: **end while**
7: $\boldsymbol{z} \leftarrow (z_0, z_1 - u, z_2 - u, \ldots, z_m - u - v)$ ▷ *Reverse line 3*
8: **if** $\Delta_1 < \epsilon$ **then** $\boldsymbol{z} \leftarrow (z_0, z_2, z_3, \ldots, z_m)$ ▷ *Ensure* $\Delta_1 \geq \epsilon$
9: **if** $\Delta_m < \epsilon$ **then** $\boldsymbol{z} \leftarrow (z_0, z_1, \ldots, z_{m-2}, z_m)$ ▷ *Ensure* $\Delta_m \geq \epsilon$

first regularising Λ to a price function Λ^ϵ with ϵ-separation for some $\epsilon > \gamma$, and only then applying the theorem. In this section we quantify the gain of going from $\gamma = 0$ to ϵ, and then derive our main payoff bound by tuning ϵ.

The regulariser Λ^ϵ is constructed by Algorithm 5.1. The key idea of the algorithm, implemented by lines 4–6, is to iteratively remove the smallest fluctuation from $\boldsymbol{z}$. This process is illustrated by Figure 5.4. The solid line shows a segment of the logprice function before regularisation. The logprice difference between the two open circles is too small, i.e. $< 2\epsilon$. The dashed line is the logprice function resulting from fluctuation removal. The other lines of the algorithm establish ϵ-separation at the boundaries of Λ.

For any sequence $\boldsymbol{z} = z_0, \ldots, z_m$ we abbreviate the terms in the bound of Theorem 5.3.5 that depend on $\boldsymbol{z}$ by defining $\vec{g} = g_1, \ldots, g_m$ and G by

$$g_i \coloneqq (z_i - z_{i-1})_+ - \ell(\Delta_i) \qquad \text{and} \qquad G \coloneqq \sum_{1 \leq i \leq m} g_i.$$

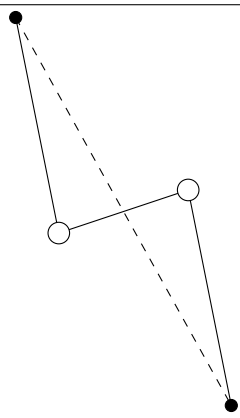
We first study the effect of a single execution of lines 4–6.

5.3.6. Lemma. *Let $\boldsymbol{z}^\circ$ and $\boldsymbol{z}^\dagger$ be the sequences before and after line 5. Then*

$$G^\dagger - G^\circ \;\geq\; (m^\circ - m^\dagger) \min\{0, \ell(2\epsilon) - \epsilon\}$$

Proof. Let i be the index of the minimal Δ_i°. Let $l = \Delta_{i-1}^\circ$, $c = \Delta_i^\circ$ and $r = \Delta_{i+1}^\circ$, so that $\Delta_{i-1}^\dagger = l + r - c$ and $2\epsilon > c \leq l, r$. By definition

Figure 5.4 Regularisation imposed by the ϵ-pruning Algorithm 5.1



$G^\dagger - G^\circ$ equals

$$\begin{aligned}\big(l+r-c-\ell(l+r-c)\big)-\big(l+r-\ell(l)-\ell(c)-\ell(r)\big) &\quad \text{if } z_{i-1}\le z_i, \text{ or}\\ \big(-\ell(l+r-c)\big)-\big(c-\ell(l)-\ell(c)-\ell(r)\big) &\quad \text{if } z_{i-1}\ge z_i.\end{aligned}$$

In either case $G^\dagger - G^\circ$ simplifies to $-c-\ell(l+r-c)+\ell(l)+\ell(c)+\ell(r)$. Since ℓ is concave, the worst-case values for l and r are c. For the same reason, the worst-case value for c is either 0 or 2ϵ. Finally

$$G^\dagger - G^\circ \;\ge\; 2\ell(c)-c \;\ge 2\min\{\ell(0),\ell(2\epsilon)-\epsilon\} \;\ge 2\min\{0,\ell(2\epsilon)-\epsilon\}$$

since ℓ is nonnegative. □

Now fix $\epsilon \ge 0$. Let $z^\epsilon = z_0^\epsilon, z_1^\epsilon, \ldots, z_{m^\epsilon}^\epsilon$ be the result of applying Algorithm 5.1 with parameter ϵ to the sequence z of local extrema of Λ, and let Λ^ϵ be any continuous function with local extrema z^ϵ. By construction Λ^ϵ has ϵ-separation and regularises Λ. Theorem 5.3.5 gives us a bound on the payoff in terms of Λ^ϵ. We now show how to get a bound in terms of the original Λ.

5.3.7. Theorem (Enforcing ϵ-Separation). *For all $\epsilon \ge 0$ such that $\ell(2\epsilon) < \epsilon$*

$$G^\epsilon - G \;\ge\; \big(m-m^\epsilon\big)\big(\ell(2\epsilon)-\epsilon\big) - 2\ell(2\epsilon).$$

Proof. Let $z^+, z^\star, z^-$ be the sequences after lines 3, 6 and 7 of Algorithm 5.1. Thus the algorithm produces (denoted $\to$) in order

$$z \;\to\; z^+ \;\to\; z^\star \;\to\; z^- \;\to\; z^\epsilon.$$

with numbers of blocks $m = m^+ \geq m^\star = m^- \geq m^\epsilon$. By Lemma 5.3.6 $G^\star - G^+ \geq \big(m^+ - m^\star\big)\big(\ell(2\epsilon) - \epsilon\big)$. It thus remains to show that

$$(G^+ - G) + (G^- - G^\star) + (G^\epsilon - G^-) \;\geq\; (m^\star - m^\epsilon)\big(\ell(2\epsilon) - \epsilon\big) - 2\ell(2\epsilon).$$

We have $G^+ - G = g_1^+ - g_1 + g_{m^+}^+ - g_m$, $G^- - G^\star = g_1^- - g_1^\star + g_{m^-}^- - g_{m^-}^\star$ and hence $G^\epsilon - G^-$ equals

$$\begin{cases} g_1^\epsilon - g_1^- - g_2^- & \text{if } \Delta_1^- < \epsilon, \\ 0 & \text{otherwise,} \end{cases} + \begin{cases} g_{m^\epsilon}^\epsilon - g_{m^-}^- - g_{m^- - 1}^- & \text{if } \Delta_{m^-}^- < \epsilon, \\ 0 & \text{otherwise.} \end{cases}$$

These three expressions are symmetric in the first and last element of the sequences concerned. The contributions of the first elements are

$$g_1^+ - g_1 \;=\; u_+ - \ell\big(\Delta_1 + |u|\big) + \ell(\Delta_1), \tag{5.3}$$

$$g_1^- - g_1^\star \;=\; -u_+ - \ell(\Delta_1^-) + \ell\big(\Delta_1^- + |u|\big), \tag{5.4}$$

$$g_1^\epsilon - g_1^- - g_2^- \;=\; -\Delta_1^- + \ell(\Delta_1^-) + \ell(\Delta_2^-) - \ell(\Delta_2^- - \Delta_1^-). \tag{5.5}$$

If $\Delta_1^- \geq \epsilon$ then no element is dropped in line 8. The sum of (5.3) and (5.4) equals

$$-\ell\big(\Delta_1 + |u|\big) + \ell(\Delta_1) - \ell(\Delta_1^-) + \ell\big(\Delta_1^- + |u|\big) \;\geq\; -\ell(2\epsilon).$$

Since ℓ increases the last two terms are positive and can be dropped from the bound; the remaining expression is increasing in Δ_1 by concavity of ℓ and is decreasing in $|u|$. Substitute the worst-case values $\Delta_1 = 0$ and $|u| = 2\epsilon$.

If on the other hand $\Delta_1^- < \epsilon$ then one element was dropped in line 8. In this case the sum of (5.3)–(5.5) equals

$$-\ell\big(\Delta_1 + |u|\big) + \ell\big(\Delta_1^- + |u|\big) + \ell(\Delta_1) - \Delta_1^- + \ell(\Delta_2^-) - \ell(\Delta_2^- - \Delta_1^-),$$

which is bounded below by $-\Delta_1^-$ as follows. First cancel the first two terms and the last two terms since ℓ is increasing and $0 \leq \Delta_1 \leq \Delta_1^-$. Since ℓ is nonnegative, omit the third term as well. Then use $-\Delta_1^- \;\geq\; -\epsilon \;=\; \big(\ell(2\epsilon) - \epsilon\big) - \ell(2\epsilon)$.

The bound for the contribution of the final elements is analogous. In each case, a dropped intermediate elements contributes at most $\ell(2\epsilon) - \epsilon$, while the borders lose at most $\ell(2\epsilon)$ each. □

We now put everything together, and in particular we optimise the value of ϵ.

5.3.8. Theorem (Payoff Bound). *Fix logprice functions Λ and Λ', the latter with associated $\boldsymbol{z}'$, m' and $\boldsymbol{\Delta}'$ as in Definition 5.3.3. If $\Lambda' \preccurlyeq \Lambda$ then*

$$\pi{*}\Lambda \;\geq\; \sum_{1\leq i\leq m'} (z'_i - z'_{i-1})_+ - \sum_{1\leq i\leq m'} \ell(\Delta'_i) - (m'-1)c_\pi - \ln 2 - 2\epsilon_\pi,$$

where ϵ_π is the unique solution to $\pi(2\epsilon) = \frac{1}{\mathrm{e}^\epsilon - 1}$, and $c_\pi = -\ln(1-\mathrm{e}^{-\epsilon_\pi})$.

Proof. For each $\epsilon \geq 0$ with $\ell(2\epsilon) < \epsilon$

$$\begin{aligned}
\pi{*}\Lambda \;&\geq\; \pi{*}\Lambda' \;\geq\; \pi{*}\Lambda^\epsilon \;\geq\; G^\epsilon + (m^\epsilon - 1)\ln(1-\mathrm{e}^{-\epsilon}) - \ln 2 \\
&\geq\; G' + (m^\epsilon - 1)\ln(1-\mathrm{e}^{-\epsilon}) + (m' - m^\epsilon)\big(\ell(2\epsilon) - \epsilon\big) - 2\ell(2\epsilon) - \ln 2 \\
&\geq\; G' + (m'-1)\min\big\{\ln(1-\mathrm{e}^{-\epsilon}), \ell(2\epsilon) - \epsilon\big\} - 2\ell(2\epsilon) - \ln 2.
\end{aligned}$$

The inequalities are twice Theorem 5.3.2, then Theorem 5.3.5, then Theorem 5.3.7. To complete the proof we set ϵ to equalise the arguments of the minimum. □

Typical values for ϵ_π and c_π are shown in Table 5.1.

5.4 Implementation

The following algorithm implements the strategy π. For arbitrary prior densities it runs in $O(T^2)$ time. For exponential priors, we reduce the running time to $O(T)$. The key to efficiency is the independent product form of π, which renders the *last* switching price a sufficient statistic.

For concreteness, we measure discrete time in days. As its data structure, the algorithm maintains a set of bank accounts. Each bank account has a *balance*, a *type* that is either A or B, and a *birthday*. The balance of type A accounts is measured in shares, whereas that of type B accounts is measured in cash.

On day zero the initial unit cash is divided evenly into two bank accounts: one account of type B with half a unit of cash, and one account of type A with $\frac{1}{2}\exp(-\Lambda(0))$ shares, i.e. half a unit of cash worth of shares at the initial logprice.

The algorithm then proceeds as follows. Each day $t = 1, 2, \ldots$ the new price $\Lambda(t)$ is announced. The algorithm creates a single new bank

account with birthday t. If $\Lambda(t-1) \leq \Lambda(t)$, then the new account is of type B, and a portion of the shares in existing accounts of type A is sold to fill it with cash. On the other hand if $\Lambda(t-1) \geq \Lambda(t)$, then a new account of type A is endowed with shares by investing a fraction of the capital of existing accounts of type B. In either case, the amount traded reestablishes the following invariant. At the end of day t:

- Each account of type A that was created with c shares on birthday i has balance $c \int_\lambda^\infty \pi(h)\,\mathrm{d}h$, where $\lambda = \max_{i \leq j \leq t} \Lambda|_i^j$.
- Each account of type B that was created with capital c on birthday i has balance $c \int_\lambda^\infty \pi(h)\,\mathrm{d}h$, where $\lambda = \max_{i \leq j \leq t} -\Lambda|_i^j$.

To see how this works, consider an A-type account with birthday i and initial balance c, and assume that the invariant was maintained at the end of day $t-1$. First, note that it can only become violated if the maximum changes, that is, if $\Lambda|_i^t$ exceeds the previous maximum $\lambda = \max_{i \leq j < t} \Lambda|_i^j$. Then the balance still is $c \int_\lambda^\infty \pi(h)\,\mathrm{d}h$ but should become $c \int_{\Lambda|_i^t}^\infty \pi(h)\,\mathrm{d}h$. The fraction

$$1 - \frac{\int_{\Lambda|_i^t}^\infty \pi(h)\,\mathrm{d}h}{\int_\lambda^\infty \pi(h)\,\mathrm{d}h} = \frac{\int_\lambda^{\Lambda|_i^t} \pi(h)\,\mathrm{d}h}{\int_\lambda^\infty \pi(h)\,\mathrm{d}h} = \pi\big(H \leq \Lambda|_i^t \big| H \geq \lambda\big) \tag{5.6}$$

of the balance must be sold to reestablish the invariant, and the resulting cash is transferred to the new account. Note that we only query π via its cumulative distribution function.

5.4.0.1 Complexity Analysis.

After t days, there are $t+2$ bank accounts to maintain, and each bank account potentially requires work each round. Thus, trading for T days takes $O(T^2)$ time and $O(T)$ space.

For exponential priors we can do better by *merging* several bank accounts into a single account with the sum of their balances. This is because for memoryless priors, the fraction (5.6) to be traded away does not depend on the birthday i, but only on the maximum λ, allowing us to merge bank accounts with the same maximum. Now observe that all bank accounts that are tapped to reestablish the invariant share

the same maximum afterwards, and can hence all be merged. This means that a bank account requires work at most *once*, namely when it is merged away. By maintaining two stacks of bank accounts, one for each type, each ordered by the maximum λ, the running time is brought down to $O(T)$. Since we do not know *when* merges happen, the space requirement is still $O(T)$, and the running time is *amortised* $O(1)$ per day.

5.5 Conclusion

We presented a simple online algorithm that can be applied to two-way trading, but also to prediction with expert advice, data compression and hypothesis testing (see Section 5.1.0.3). Compared to the many hedging algorithms described in the literature, our approach has two novel properties. First, the overhead of our algorithm is independent of the times at which prices are processed, and second, our bound is free of any conditions on the evolution of the price of the asset, and is parameterised either by the asset price function itself or by a regularised model of it.

The surprisingly simple implementation (Section 5.4) processes a sequence of T asset prices in $O(T^2)$ time and $O(T)$ space. The algorithm models the scale of the fluctuations of the price using a density function on $[0, \infty)$; if an exponential density is employed, the running time is reduced to $O(T)$.

Chapter 6

Hedging Structured Concepts

$$\underset{w \in \mathrm{conv}(\mathcal{C})}{\operatorname{argmin}} \quad \triangle(w \| w^{t-1}) + \eta w \cdot \ell^t$$

Abstract We develop an online algorithm called *Component Hedge* for learning structured concept classes when the loss of a structured concept sums over its components. Example classes include paths through a graph (composed of edges) and partial permutations (composed of assignments). The algorithm maintains a parameter vector with one non-negative weight per component, which always lies in the convex hull of the structured concept class. The algorithm predicts by decomposing the current parameter vector into a convex combination of concepts and choosing one of those concepts at random. The parameters are updated by first performing a multiplicative update and then projecting back into the convex hull. We show that Component Hedge has optimal regret bounds for a large variety of structured concept classes.

6.1 Introduction

We develop online learning algorithms for structured concepts that are composed of components. For example, sets are composed of elements, permutations of individual assignments, trees have edges as components, etc. The number of components d is considered small, but the number of structured concepts D built from the components is typically exponential in d.

Our algorithms address the following online prediction problem. In each trial the algorithm first produces a concept from the structured class by choosing a concept probabilistically based on its current parameters. It then observes the loss of each concept. Finally, it prepares for the next trial by updating its parameters by incorporating the losses. Since the algorithm "hedges" by choosing the structured concept probabilistically, we analyse the expected loss incurred in each trial. The goal is to develop algorithms with small regret, which is the total expected loss of the online algorithm minus the loss of the best structured concept in the class chosen in hindsight.

We now make a key simplifying assumption on the loss: We assume that the loss of a structured concept in each trial is always the sum of the losses of its components and that the component losses always have range $[0, 1]$. Thus if the concepts are k-element sets chosen out of n elements, then in each trial each element is assigned a loss in $[0, 1]$ and the loss of any particular k-set is simply the sum of the losses of its elements. Similarly for trees, a loss in $[0, 1]$ is assigned to each edge of the graph and the loss of a tree is the sum of the losses of its edges.

We will show that with this simplifying assumption we still have rich learning problems that address a variety of new settings. We give efficient algorithms (i.e. polynomial in d) that serve as an entry point for considering more complex losses in the future.

Perhaps the simplest approach to learn structured concept classes online is the Follow the Perturbed Leader (FPL) algorithm [92]. FPL adds a random perturbation to the cumulative loss of each individual component, and then plays the structured concept with minimal perturbed loss. FPL is widely applicable, since efficient combinatorial optimisation algorithms exist for a broad range of concept classes. Unfortunately, the loss range of the structured concepts enters into the regret bounds that we can prove for FPL. For example, for k-sets the

loss range is $[0, k]$ because each set contains k elements, for permutations the loss range is $[0, n]$ because each permutation is composed of n assignments, etc.

A second simple approach for learning well compared to the best structured concept is to run the Hedge algorithm of [59] with one weight per structured concept. The original algorithm was developed for the so-called expert setting, which in the context of this chapter corresponds to learning with sets of size one. To apply this algorithm to our setting, the experts are chosen as the structured concepts in the class we are trying to learn. In this chapter we call this algorithm *Expanded Hedge* (EH). It maintains its uncertainty as a probability distribution over all structured concepts and the weight W_C of concept C is proportional to $\exp(-\eta \ell(C))$, where $\ell(C)$ is the total loss of concept C incurred so far and η is a non-negative learning rate.

There are two problems with EH. First, there are exponentially many weights to maintain. However our simplifying assumption assures that $\ell(C)$ is a sum over the losses of the component of C. This implies that W_C is proportional to a product over the components of the structured concept C and this fact can be exploited to still achieve efficient algorithms in some cases. More importantly however, like for FPL, the loss range of the structured concepts usually enters into the best regret bounds that we can prove.

Learning with structured concepts has also been dealt with recently in the bandit domain [26]. However all of this work is based on EH and contains the additional range factors.

Our contribution Our new method, called *Component Hedge* (CH), avoids the additional range factors altogether. Each structured concept C is identified with its incidence vector in $\{0,1\}^d$ indicating which components are used. The parameter space of CH is simply the convex hull of all concepts in the class $\mathcal{C}$ to be learned. Thus, whereas EH maintains a weight for each structured concept, CH only maintains a weight for each component. The current parameter vector represents CH's first-order "uncertainty" about the quality of each concept. The value of parameter i represents the *usage* of component i in the next prediction. The usages of the components are updated in each trial by incorporating the current losses, and if the usage vector leaves the hull, then it is projected back via a relative entropy projection. The key trick

to make this projection efficient is to find a representation of the convex hull of the concepts as a convex polytope with a number of facets that is polynomial in d. We give many applications where this is possible.

We clearly champion the Component Hedge algorithm in this chapter because we can prove regret bounds for this algorithm that are tight within constant factors for many structured concept classes. Also it is trivial to enhance CH with a variety of "share updates" that make it robust in the case when the best comparator changes over time [80, 19].

Two instances of CH have appeared before even though this name was not used: learning with k-sets [185] and learning with permutations [77]. The same polytope we use for paths was also employed in [5] for developing online algorithms for the bandit setting. They avoid the projection step altogether by exploiting a barrier function. The contribution of this chapter is to clearly formulate the general methodology of the Component Hedge algorithm and give many more involved combinatorial examples. In the case of permutations we also show how the method can be used to learn truncated permutations. Also in earlier work [173] it was pointed out that the Expanded Hedge algorithm can be simulated efficiently in many cases. In particular, the concept class of paths in a directed graph was introduced. However, good bounds were only achieved in very special cases. In this chapter we show that CH essentially is optimal for the path problem.

Outline We give the basic setup for the structured prediction task, introduce CH and prove its general regret bound in Section 6.2. We then turn to a list of applications in Section 6.3: vanilla experts, k-sets, permutations, paths, undirected and directed spanning trees. For each structured concept class we discuss efficient implementation of CH, and derive expected regret bounds for this algorithm. Then in Section 6.4 we provide matching lower bounds for all examples, showing that the regret of CH is optimal within a constant factor. In Section 6.5 we compare CH to the existing algorithms EH and FPL. We observe that the best general regret bounds for each algorithm exceed that of CH by a significant range factor. We show that the bounds for these other algorithms can be improved to closely match those of CH whenever the the so-called *unit rule* holds for the algorithms and class. This means any loss vector $\ell \in [0,1]^d$ can be split into up to d scaled unit loss vectors $\ell_i\, e_i$ and processing these in separate trials always incurs at least

as much loss. Unfortunately, for most pairing of the algorithms CH and FPL with the classes we consider in this chapter, we have explicit counter examples to the unit rule. Finally, Section 6.6 concludes with a list of open problems.

6.2 Component Hedge

Prediction task We consider sequential prediction [75, 25] based on a structured concept class [92, 26]. Fix a set of concepts $\mathcal{C} \subseteq \{0,1\}^d$ of size $D = |\mathcal{C}|$. For example $\mathcal{C}$ could consist of the incidence vectors of subsets of k out of n elements (then $D = \binom{n}{k}$ and $d = n$), or the adjacency matrices of undirected spanning trees on n elements (then $D = n^{n-2}$ and $d = n(n-1)/2$).

Our online learning protocol proceeds in trials. At trial t, we have to produce a single concept $C^t \in \mathcal{C}$. Then a loss vector $\ell^t \in [0,1]^d$ is revealed, and we incur loss given by the dot product $C^t \cdot \ell^t$. Although each component suffers loss at most 1, a concept may suffer loss up to $U := \max_{C \in \mathcal{C}} |C|$. We allow randomised algorithms. Thus the expected loss of of the algorithm at trial t is $\mathbb{E}[\mathbf{C}^t] \cdot \ell^t$, where the expectation is over the internal randomisation of the algorithm. Our goal is to minimise our (expected) *regret* after T trials

$$\sum_{t=1}^{T} \mathbb{E}[\mathbf{C}^t] \cdot \ell^t - \min_{C \in \mathcal{C}} \sum_{t=1}^{T} C \cdot \ell^t.$$

That is, the difference between our cumulative expected loss and the loss of the best concept in hindsight.

Note that the ith component of $\mathbb{E}[\mathbf{C}^t]$ is the probability that component i is "used in" concept $\mathbf{C}^t$. We therefore call $\mathbb{E}[\mathbf{C}^t]$ the *usage vector*. This vector becomes the internal parameter of our algorithm. The set of all usage vectors is the convex hull of the concepts.

6.2.1 Component Hedge

Two instances of CH appeared before in the literature [77, 185]. Here we give the algorithm in its general form, and prove a general regret bound. The algorithm CH maintains its uncertainty about the best structured concept as a usage vector w^t in $\mathrm{conv}(\mathcal{C}) \subseteq [0,1]^d$, the convex

Table 6.1 Example structured concept classes

Case	U	D	d
Experts	1	n	n
k-Sets	k	$\binom{n}{k}$	n
Permutations	n	$n!$	n^2
Paths (from source to sink)	$n+1$	$n! \cdot \mathrm{e} - o(1)$	$n(n+1)+1$
Undirected spanning trees	$n-1$	n^{n-2}	$n(n-1)/2$
Directed spanning trees	$n-1$	n^{n-2}	$(n-1)^2$

hull of the concepts $\mathcal{C}$. The initial weight w^0 is typically the usage of the uniform distribution on concepts. CH predicts in trial t by decomposing w^{t-1} into a convex combination[1] of the concepts $\mathcal{C}$, then sampling $\mathbf{C}^t$ according to its weight in that convex combination. The expected loss of CH is thus $w^{t-1} \cdot \ell^t$. The updated weight w^t is obtained by trading off the relative entropy with the linear loss:

$$w^t := \operatorname*{argmin}_{w \in \mathrm{conv}(\mathcal{C})} \triangle(w \| w^{t-1}) + \eta w \cdot \ell^t,$$

where the relative entropy is defined by

$$\triangle(w \| v) = \sum_{i \in [d]} \left(w_i \ln \frac{w_i}{v_i} + v_i - w_i \right).$$

It is easy to see that this update can be split into two steps: an unconstrained update followed by relative entropy projection into the convex hull:

$$\begin{aligned} \widehat{w}^t &:= \operatorname*{argmin}_{w \in \mathbb{R}^d} \triangle(w \| w^{t-1}) + \eta w \cdot \ell^t \\ w^t &:= \operatorname*{argmin}_{w \in \mathrm{conv}(\mathcal{C})} \triangle(w \| \widehat{w}^t). \end{aligned}$$

It is easy to see that $\widehat{w}_i^t = w_i^{t-1} \mathrm{e}^{-\eta \ell_i^t}$, that is, the old weights are simply scaled down by the exponentiated losses. The result of the relative

[1] This decomposition usually is far from unique.

entropy projection w^t unfortunately does not have a closed form expression.

For CH to be efficiently implementable, the hull has to be captured by polynomial in d many constraints. This will allow us to efficiently decompose any point in the hull as a convex combination of at most $d+1$ concepts. The trickier part is to efficiently implement the projection step. For this purpose one can use generic convex optimisation routines. For example this was done in the context of implementing the entropy regularised boosting algorithm [186]. We proceed on a case by case basis and often develop iterative algorithms that locally enforce constraints and do multiple passes over all constraints. See Table 6.1 for a list of structured concept classes we consider in this chapter.

6.2.2 Regret Bounds

As in [77], the analysis is split into two steps parallelling the two update steps. Essentially the unnormalised update step already gives the regret bound and the projection step does not hurt. For any usage vector $w^{t-1} \in \mathrm{conv}(\mathcal{C})$, loss vector $\ell^t \in \{0,1\}^d$ and any comparator concept C,

$$\begin{aligned}(1-\mathrm{e}^{-\eta})w^{t-1}\cdot\ell^t \;&\leq\; \underbrace{\triangle(C\|w^{t-1})-\triangle(C\|\widehat{w}^t)+\eta\; C\cdot\ell^t}_{\sum_i w_i^{t-1}(1-\mathrm{e}^{-\eta\ell_i^t})}\\ &\leq\; \triangle(C\|w^{t-1})-\triangle(C\|w^t)+\eta\; C\cdot\ell^t\end{aligned}$$

The first inequality is obtained by bounding the exponential using the inequality $1-\mathrm{e}^{-\eta x}\geq(1-\mathrm{e}^{-\eta})x$ for $x\in[0,1]$ as done in [108]. The second inequality is an application of the Generalised Pythagorean Theorem [81], using the fact that w^t is a Bregman projection of $\widehat{w}^t$ into the convex set $\mathrm{conv}(\mathcal{C})$, which contains C. We now sum over trials and obtain, abbreviating $\ell^1+\ldots+\ell^T$ to $\ell^{\leq T}$,

$$(1-\mathrm{e}^{-\eta})\sum_{t=1}^{T} w^{t-1}\cdot\ell^t \;\leq\; \triangle(C\|w^0)-\triangle(C\|w^T)+\eta C\cdot\ell^{\leq T}.$$

Recall that $w^{t-1}\cdot\ell^t$ equals the expected loss $\mathbb{E}[\mathbf{C}^t]\cdot\ell^t$ of CH in trial t. Also, relative entropies are nonnegative, so we may drop the second one, giving us the following bound on the total loss of the algorithm:

$$\sum_{t=1}^{T}\mathbb{E}[\mathbf{C}^t]\cdot\ell^t \;\leq\; \frac{\triangle(C\|w^0)+\eta C\cdot\ell^{\leq T}}{1-\mathrm{e}^{-\eta}}.$$

To proceed we have to expand the prior w^0. We consider the *symmetric balanced* case, i.e. where the concept class is invariant under permutation of the components, and every concept uses exactly U components. Paths may have different lengths and hence do not satisfy these requirements. All other examples from Table 6.1 do. In this balanced symmetric case we take w^0 to be the usage of the uniform distribution on concepts, satisfying $w_i^0 = U/d$ for each component i. It follows that $\triangle(C\|w^0) = U\ln(d/U)$, because any comparator C is a $0/1$ vector that also uses exactly U components.

Let $\ell^\star$ denote $\min_{C\in\mathcal{C}} C\cdot\ell^{\leq T}$, the loss of the best concept in hindsight. Then by choosing $\eta = \sqrt{\frac{2U\ln(d/U)}{\ell^\star}}$ as a function of $\ell^\star$, we obtain the following general expected regret bound for CH:

$$\mathbb{E}\left[\ell_{\mathrm{CH}}\right] - \ell^\star \;\leq\; \sqrt{2\ell^\star U\ln(d/U)} + U\ln(d/U). \tag{6.1}$$

The best-known general regret bounds for Expanded Hedge [59] and Follow the Perturbed Leader [84] are:

$$\mathbb{E}\left[\ell_{\mathrm{EH}}\right] - \ell^\star \;\leq\; \sqrt{2\ell^\star U\ln D} + U\ln D \tag{6.2}$$

$$\mathbb{E}\left[\ell_{\mathrm{FPL}}\right] - \ell^\star \;\leq\; \sqrt{4\ell^\star Ud\ln d} + 3Ud\ln d \tag{6.3}$$

where $D = |\mathcal{C}|$. Specific values for U, D and d in each application are listed in Table 6.1. We remark that if only an upper bound $\hat{\ell} \geq \ell^\star$ is available, then we can still tune η as a function of $\hat{\ell}$ to achieve these bounds with $\hat{\ell}$ under the square roots instead of $\ell^\star$. Moreover, standard heuristics can be used to tune η "online" when no good upper bound on $\ell^\star$ is given, which increase the expected regret bounds by at most a constant factor. (e.g. [28, 84]).

We are not concerned with small multiplicative constants (e.g. 2 vs 4), but the gap between (6.1) and both (6.2) and (6.3) is significant. To compare, observe that $\ln D$ is of order $U\ln d$ in all our applications. Thus, the EH regret bound is worse by a factor $\sqrt{U}$, while FPL is worse by a bigger factor $\sqrt{d}$. Moreover, in Section 6.4 we show for the covered examples that our expected regret bound (6.1) for CH is optimal up to constant scaling.

Some concept classes have special structure that can be exploited to improve the regret bounds of FPL and EH down to that of CH. We consider one such property, called the *unit rule* in Section 6.5.

6.3 Applications

We consider the following structured concept classes: experts, k-sets, truncated permutations, source-sink paths, and both undirected and directed spanning trees. In each case we discuss implementation of CH and obtain a regret bound. Matching lower bounds are presented in Section 6.4.

6.3.1 Experts

The most basic example is the vanilla expert setting. In this case, the set of "structured" concepts equals the set of n standard basis vectors in $\mathbb{R}^n$. We will see that in this case Component Hedge see gracefully degrades to the original Hedge algorithm. First, the parameter spaces of both algorithms coincide since the convex hull of the basis vectors equals the probability simplex. Second, the predictions coincide since a vector in the probability simplex decomposes uniquely into a convex combination of basis vectors. Third, the parameter updates are the same, since the relative entropy projection of a non-negative weight vector into the probability simplex amounts to re-normalising to unity.

In fact on this simple task CH, EH and FPL each coincide with Hedge. For CH and EH this is obvious. For FPL this fact was observed in [102, 91] by using log-of-exponential perturbations instead of exponential perturbations used in the original paper [92]. Thus, we obtain following regret bound for all algorithms:

$$\mathbb{E}\left[\ell_{\mathrm{CH}}\right] - \ell^\star \;\leq\; \sqrt{2\ell^\star \ln n} + \ln n.$$

6.3.2 k-sets

The problem of learning with sets of k out of n elements was introduced in [185] and applied to online Principal Component Analysis (PCA). Their algorithm is an instance of CH, and we review it here. The convex hull of k-sets equals the set of $w \in \mathbb{R}^n_+$ that satisfy the following constraints:

$$w_i \leq 1 \quad \text{for all } i \in [n] \qquad\qquad \text{and} \qquad\qquad \sum_{i=1}^{n} w_i = k. \tag{6.4}$$

Relative entropy projection into this polytope amounts to renormalising the sum to k, followed by redistributing the mass of the components that exceed 1 over the remaining components so that their ratios are preserved. Finally, each element of the convex hull of sets can be greedily decomposed into a convex combination of n k-sets by iteratively removing sets in the convex combination while always setting the coefficient of the new set as high as possible. Both projection and decomposition take $O(n^2)$ time [185].

Regret bound By (6.1), the regret of CH on sets is

$$\mathbb{E}\left[\ell_{\mathrm{CH}}\right] - \ell^\star \;\leq\; \sqrt{2\ell^\star k \ln(n/k)} + k\ln(n/k).$$

We give a matching lower bound in Section 6.4.

6.3.3 Truncated Permutations

The second instantiation of CH that has appeared is the problem of permutations [77]. Here we consider a slightly generalised task: *truncated permutations* of k out of n elements. A truncated permutation fills k slots with distinct elements from a pool of n elements. Equivalently, a truncated permutation is a maximal matching in the complete bipartite graph between $[k]$ and $[n]$. Truncated permutations extend k-sets by linearly ordering the selected k elements.

Results to search queries are usually in the form of a truncated permutation; of all n existing documents, only the top k are displayed in order of decreasing relevance. Predicting with truncated permutations is thus a model for learning the best search result.

Matching polytope We write $i \leftarrow j$ for the component that assigns item j to slot i. Now the convex hull of truncated permutations consists of all $w \in \mathbb{R}_+^{k\times n}$ (see [161, Corollary 18.1b]) satisfying the following k *row* (left) and n *column* (right) constraints:

$$\sum_{j\in[n]} w_{i\leftarrow j} = 1 \quad \text{for all } i \in [k] \quad \text{and} \quad \sum_{i\in[k]} w_{i\leftarrow j} \leq 1 \quad \text{for all } j \in [n]. \tag{6.5}$$

Relative entropy projection The relative entropy projection of $\widehat{w}$ into the convex hull of truncated permutations $w = \operatorname{argmin}_{w \text{ s.t. } (6.5)} \triangle(w\|\widehat{w})$ has no closed form solution. By convex duality (details are given in Appendix 6.B.1), $w_{i\leftarrow j} = \widehat{w}_{i\leftarrow j} e^{-\lambda_i - \mu_j}$, where λ_i and μ_j are the Lagrange multipliers associated to the row and column constraints (6.5), which minimise

$$\sum_{i\in[k]\,;\,j\in[n]} \widehat{w}_{i\leftarrow j} e^{-\lambda_i-\mu_j} + \sum_{i\in[k]} \lambda_i + \sum_{j\in[n]} \mu_j.$$

under the constraint that $\mu \geq \mathbf{0}$. This dual problem, which has $2n$ variables and n constraints, may be optimised directly using numerical convex optimisation software. Another approach is to iteratively reestablish each violated constraint beginning from $\mu = \mathbf{0}$ and $\lambda = \mathbf{0}$. In full permutation case ($k = n$), this process is called *Sinkhorn balancing*. It is known to converge to the optimum, see [77] for an overview of efficiency and convergence results of this iterative method.

Decomposition Our decomposition algorithm for truncated permutations interpolates between the decomposition algorithms used for k-sets and full permutations [185, 77]. Assume w lies in the hull of truncated permutations, i.e. the constraints (6.5) are satisfied. To measure progress, we define a score $s(w)$ as the number of zero components in w plus the number of column constraints that are satisfied with equality.

Our algorithm maintains a truncated permutation C that satisfies the following invariant: C hits all columns whose constraints are satisfied with equality by w, and avoids all components with weight zero in w. Such a C can be established in time $O(k^2 n)$ using augmenting path methods (see [161, Theorem 16.3]).

Let l be the minimum weight of the components used by C, and let h be the maximum column sum of the columns untouched by C. So by construction $h < 1$. If $l = 1$ then $w = C$ and we are done. Otherwise, let $\alpha = \min\{l, 1-h\}$, and set $w' = (w - \alpha C)/(1-\alpha)$. It is easy to see that the vector w' satisfies (6.5), and that $s(w') > s(w)$. It is no longer the case that C satisfies the invariant w.r.t. w'. However, we may compute a weight k matching C' that satisfies the invariant by executing at most $s(w') - s(w)$ many augmenting path computations, which each cost $O(kn)$ time. We describe how this works below. After that we simply recurse on w' and C'. The resulting convex combination is αC plus $(1-\alpha)$ times the result of the recursion.

The number of iterations is bounded by the score $s(w)$, which is at most kn. Thus, the total running time is $O(k^2n^2)$.

We now show that C can be improved to C' satisfying the invariant by a single augmenting path computation per violated requirement. Let C^* be a size k matching satisfying the invariant for w'. Such a matching always exists because w' lies in the matching polytope. Let $j \in [n]$ be a problematic column, i.e. either C matches j to a row i but $w'_{i \leftarrow j} = 0$, or C does not match j while its column constraint is tight for w'. From j, alternately follow edges from C and C^*. Since C and C^* are both matchings, this can not lead to a cycle, so it must lead to a path. Since all rows are matched, this path must end at a column. The path can not end at a column whose constraint is forced in both C and C^*. So it must end at a column whose constraint is not tight. Incorporating this augmenting path into C corrects the violated requirement without creating any new violations.

Regret bound By (6.1), the regret of CH on truncated permutations is

$$\mathbb{E}\left[\ell_{\mathrm{CH}}\right] - \ell^\star \;\leq\; \sqrt{2\ell^\star k \ln n} + k \ln n.$$

We obtain a matching lower bound in Section 6.4.

6.3.4 Paths

The online shortest path problem was considered by [173, 92], and by various researchers in the bandit setting (see e.g. [26, 5] and references therein). We develop expected regret bounds for CH for the "full information setting". Our regret bound improves the bounds given in [173, 92] which have the additional range factors in the square root.

Consider the a directed graph on the set of nodes $[n] \cup \{s, t\}$. Each trial we have to play a walk from the source node s to the sink node t. As always, our loss is given by the sum of the losses of the edges that our walk traverses. Since each edge loss is nonnegative (it lies in $[0, 1]$ by assumption) it is never beneficial to visit a node more than once. Thus w.l.o.g. we restrict attention to paths.

As an example, consider the full directed graph on $[n] \cup \{s, t\}$. Paths of length $k + 1$ through this graph use k distinct internal nodes in order, and therefore are in 1-1 correspondence with truncated permutations

of size k. Paths thus generalise truncated permutations by allowing all lengths simultaneously.

Unit flow polytope To implement CH efficiently, we have to succinctly describe the convex hull of paths. Unfortunately, we can not hope to write down linear constraints that capture the convex hull *exactly*. For if we could, then we could solve the *longest path* problem, which is known to be NP complete, by linear programming. Fortunately, there is a slight relaxation of the convex hull of paths that is describable by few constraints, namely the polytope of so-called unit flows. Even better, we will see that this relaxation does not hurt predictive performance at all.

A *unit flow* $w \in \mathbb{R}_+^d$ is described by the following constraints:

$$1 = \sum_{j \in [n]+t} w_{s,j} \quad \text{and} \quad \sum_{j \in [n]+s} w_{j,i} = \sum_{j \in [n]+t} w_{i,j} \quad \text{for each } i \in [n]. \tag{6.6}$$

We think of $w_{i,j}$ as describing the amount of flow from node i to j. The left constraint ensures that one unit of flow leaves the source s. The right constraint enforces that at internal nodes inflow equals outflow. It easily follows that one unit of flow enters the sink t.

The unit flow polytope is not bounded, but it has the right "bottom". Namely, the vertices of the unit flow polytope are the s-t paths, see [161, Section 10.3]. The unit flow polytope is the Minkowski sum of the convex hull of s-t paths and the conic hull (nonnegative linear combinations) of directed cycles. Moreover, each unit flow can be decomposed into at most d paths and cycles, by iterative greedy removal of a directed cycle or paths containing the edge of least non-zero weight in time $O(n^4)$.

Since the unit flow polytope does have polynomially many constraints, we may efficiently run CH on it. Each round, it produces a flow. We then decompose this flow into paths and cycles, and throw away the cycles. We then sample a path from the remaining convex combination of paths.

Relative entropy projection To run CH, we have to compute the relative entropy projection of an arbitrary vector in $\mathbb{R}_+^d$ into the flow polytope (6.6). This is a convex optimisation problem in $d \approx n^2$ variables

with constraints. By Slater's constraint condition, we have strong duality. So equivalently, we may solve the concave dual problem, which only has $n+1$ variables and is unconstrained. The dual problem (details are given in Appendix 6.B.2) can therefore be solved efficiently by numerical convex optimisation software.

Say we want to find w, the relative entropy projection of $\widehat{w}$ into the flow polytope. Since each edge appears in exactly two constraints with opposite sign, the solution has the form $w_{i,j} = \widehat{w}_{i,j}\mathrm{e}^{\lambda_i - \lambda_j}$ for all $i, j \in [n] \cup \{s, t\}$, where λ_i is the Lagrange multiplier associated with node i (and $\lambda_t = 0$). The vector λ maximises

$$\lambda_s - \sum_{i \neq t \,;\, j \neq s} \widehat{w}_{i,j}\mathrm{e}^{\lambda_i - \lambda_j}$$

That is, we have to find a single scale factor e^{λ_i} for each node i, such that scaling each edge weight by the ratio of the factors of its nodes reestablishes the flow constraints (6.6).

We propose the following iterative algorithm. Start with all λ_i equal to zero. Then pick a violated constraint, say at node i, and reestablish it by changing its associated λ_i. That is, we execute either

$$\mathrm{e}^{\lambda_s} \leftarrow \frac{1}{\sum_{j \in [n]+t} \widehat{w}_{s,j}\mathrm{e}^{-\lambda_j}}$$

or

$$\mathrm{e}^{\lambda_i} \leftarrow \sqrt{\sum_{j \in [n]+s} \widehat{w}_{j,i}\mathrm{e}^{\lambda_j} \Big/ \sum_{j \in [n]+t} \widehat{w}_{i,j}\mathrm{e}^{-\lambda_j}} \qquad \text{for some } i \in [n].$$

In our experiments, this algorithm converges quickly. We leave its thorough analysis as an open problem.

Decomposition Find any s-t path with non-zero weights on all edges in time $O(n^2)$. Subtract that path, scaled by its minimum edge weight. This creates a new zero, maintains flow balance, and reduces the outflow of the source. After at most n^2 iterations the source has outflow zero. Discard the remaining conic combination of directed cycles. The total running time is $O(n^4)$.

Regret bound for the complete directed graph Since paths have different lengths, we aim for a regret bound that depends on the length of the comparator path. To get such a bound, we need a prior usage vector w^0 that favours shorter paths. To this end, consider the distribution $\mathbb{P}$ that distributes weight 2^{-k} uniformly over all paths of length $k \leq n$, and assigns weight 2^{-n} to the paths of length $n+1$. This assures that $\mathbb{P}$ is normalised to 1. Since there are $n!/(n-k+1)!$ paths of length k, the probability of a path P of length k equals

$$\mathbb{P}(\mathbf{P} = P) \;=\; \begin{cases} \dfrac{(n-k+1)!}{2^k n!} & \text{if } k \leq n, \\ \dfrac{1}{2^n n!} & \text{if } k = n+1. \end{cases}$$

Also, the expected path length $\mathbb{E}[\mathbf{P} \cdot \mathbf{1}]$ is $2 - 2^{-n}$. We now set $w^0 := \mathbb{E}[\mathbf{P}]$, i.e. the usage of $\mathbb{P}$. There are three kinds of edges. We have one direct edge s,t, we have $2n$ boundary edges of the form s,j or i,t, and we have $n(n-1)$ internal edges of the type i,j. A simple computation shows that their usages are (for $n \geq 3$)

$$w^0_{s,t} = \frac{1}{2}, \qquad w^0_{s,j}\,,\; w^0_{i,t} = \frac{1}{2n}, \qquad w^0_{i,j} = \frac{1 - 2^{-(n-1)}}{2n(n-1)}.$$

Let P be a comparator path of length k. If $k = 1$ then $\triangle(P \| w^0) = \ln 2$. Otherwise, still for $n \geq 3$,

$$\begin{aligned} \triangle(P\|w^0) \;&=\; -2\ln\frac{1}{2n} - (k-2)\ln\frac{1-2^{-(n-1)}}{2n(n-1)} + \mathbb{E}[\mathbf{P}\cdot\mathbf{1}] - k \\ &=\; (k-2)\ln\big(2n(n-1)\big) + 2\ln 2n + (k-2)\ln\left(1 + \frac{2^{-(n-1)}}{1-2^{-(n-1)}}\right) - \\ 2^{-n} - (k-2) \;&\leq\; k\ln 2 - (k-2)\frac{1-2^{-n+2}}{1-2^{-n+1}} + 2(k-1)\ln n \;\leq\; 2k\ln n. \end{aligned}$$

By tuning η as before, the regret of CH with prior w^0 w.r.t. a comparator path of length k is

$$\mathbb{E}\left[\ell_{\mathrm{CH}}\right] - \ell^\star \;\leq\; \sqrt{4\ell^\star k \ln n} + 2k\ln n.$$

This new regret bound improves known results in two ways. First, it does not have the range factors, which in the case of paths usually turn

out to be the diameter of the graph, i.e. the length of the longest *s*-*t* path. Second, some previous bounds only hold for acyclic graphs. Our bound holds for the complete graph.

Regret bound for an arbitrary graph We discussed the full graph as a first application of CH. For prediction on an arbitrary graphs we simply design a prior w^0 with zero usage on all edges that are not present in the graph. We could either use graph-specific knowledge, or we could use our old w^0, disable edges by setting their usage to zero, and project back into the flow polytope. Relative entropy projection never revives zeroed edges. The regret bound now obviously depends on the graph via the prior usage w^0.

6.3.4.1 Expanded Hedge and Component Hedge are Different on Paths

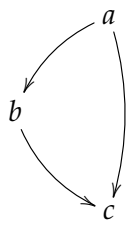
An efficient dynamic programming-based algorithm for EH was presented in [173]. This algorithm keeps one weight per edge, just like CH. These weights are updated using the *weight pushing algorithm*. This algorithm performs relative entropy projection on full distributions on paths. Like CH, weight pushing finds a weight of each node, and scales each edge weight by the ratio of its nodes weights. We now show that CH and EH are different on graphs. Consider the graph shown in Figure 6.1a. Say we use prior $\mathbb{P}$ with weight $1/2$ on both paths (a, b, c) and (a, c). Then the usages are $(1/2, 1/2, 1/2)$ for (ab, bc, ac). Now multiply edge ab by $1/3$ (that is, we give it loss $\ln 3$), and both other edges by 1 (we give them loss zero). The resulting usages of EH and CH are displayed in Table 6.1b. The usages are different, and hence, so are the expected losses. In most cases (as shown e.g. in Table 6.1c), the updated usages of CH are irrational while the prior usages and the scale factors of the update are rational. On the other hand, EH always maintains rationality.

6.3.5 Spanning Trees

Whereas paths connect the source to the sink, spanning trees connect every node to every other node. Undirected spanning trees are often used in network-level communication protocols. For example, the *Spanning Tree Protocol* (IEEE 802.1D) is used by mesh networks of Ethernet

Figure 6.1 Expanded Hedge is not Component Hedge on paths

(a) Graph



(b) Usages after update $(1/3, 1, 1)$

Case	ab, bc	ac
EH and CH prior	$1/2$	$1/2$
EH after update	$1/4$	$3/4$
CH after update	$1/3$	$2/3$

(c) Usages after update $(1/2, 1, 1)$

Case	ab, bc	ac
EH and CH prior	$1/2$	$1/2$
EH after update	$1/3$	$2/3$
CH after update	$\frac{\sqrt{17}-1}{8}$	$\frac{9-\sqrt{17}}{8}$

switches to agree on a single undirected spanning tree, and thus eliminate loops by disabling redundant links. Directed spanning trees are used for asymmetric communication, for example for streaming multimedia from a central server to all connected clients. In either case, the cost of a spanning tree is the sum of the costs of its edges.

Learning spanning trees was pioneered by [99] for learning dependency parse trees. They discuss efficient methods for parameter estimation under log-loss and hinge loss. [26] derive a regret bound for undirected spanning trees in the bandit setting. We instantiate CH to both directed and undirected trees and give the first regret bound without the range factor.

Three kinds of directed spanning trees are common. Spanning trees with a fixed root, spanning trees with a single arbitrary root, and arborescences (or spanning forests) with multiple roots. We focus on a fixed root. The other two models can be simulated by a fixed root. To simulate arborescences, add a dummy as the fixed root, and put the root selection cost of node i along the path from the dummy to i. Furthermore, to force a single root, increase the cost of all edges leaving the dummy by a fixed huge amount.

Tree polytope To characterise the convex hull of directed trees on n nodes with fixed root 1, we use a trick based on flows from [112] that makes use of auxiliary variables $f^k_{i,j}$. For $i, j, k \in [n]$ the constraints are

$$0 \leq f^k_{i,j} \leq w_{i,j}, \qquad \sum_{i,j} w_{i,j} = n - 1, \tag{6.7a}$$

and

$$\underbrace{\sum_{j \neq i} f^k_{j,i}}_{k\text{-flow into } i} + \underbrace{\mathbf{1}_{i=1}}_{k\text{-source at 1}} = \underbrace{\sum_{j \neq i} f^k_{i,j}}_{k\text{-flow out of } i} + \underbrace{\mathbf{1}_{i=k}}_{k\text{-sink at } k}. \tag{6.7b}$$

The intuition is as follows. A tree has $n-1$ edges, and every node can be reached from the root. We enforce this by having a separate flow channel f^k for each non-root node k. We place a unit of flow into this channel at the root. Each intermediate node satisfies flow equilibrium. Finally, the target node k consumes the unit of flow destined for it. The first equation ensures that each edge's usage is sufficient for the flow that traverses that edge. The undirected tree polytope is constructed based on the directed tree polytope by considering the above $w_{i,j}$ as auxiliary variables, an imposing the constraint $w_{i,j} + w_{j,i} = v_{i,j}$. Now v are the weights sought.

Relative entropy projection The relative entropy projection of $\widehat{w}$ into the convex hull of directed spanning trees $w = \operatorname{argmin}_{w \text{ s.t. } (6.7)} \triangle(w \| \widehat{w})$ has no closed form solution. By convex duality (details are given in Appendix 6.B.3), the solution satisfies

$$w_{i,j} \;=\; (n-1)\, \frac{\widehat{w}_{i,j} \mathrm{e}^{\sum_{k\neq 1} \max\{0, \mu^k_j - \mu^k_i\}}}{\sum_{i,j\neq i} \widehat{w}_{i,j} \mathrm{e}^{\sum_{k\neq 1} \max\{0, \mu^k_j - \mu^k_i\}}}, \quad f^k_{ij} = \begin{cases} w_{i,j} & \text{if } \mu^k_j > \mu^k_i, \\ 0 & \text{if } \mu^k_j < \mu^k_i, \end{cases}$$

where μ^k_i, the Lagrange multipliers associated to the flow balance constraints, maximise

$$\sum_{k\neq 1} \left(\mu^k_k - \mu^k_1\right) - (n-1) \ln\Bigg(\sum_{i,j\neq i} \widehat{w}_{i,j} \mathrm{e}^{\sum_{k\neq 1} \max\{0, \mu^k_j - \mu^k_i\}} \Bigg).$$

This unconstrained concave maximisation problem in $\approx n^2$ variables seems easier than the primal problem, which has $\approx n^3$ variables and

constraints. Note however that the objective is not differentiable everywhere. Alternatively, we may again proceed by iteratively reestablishing constraints locally, starting from some initial assignment to the dual variables μ. This approach is analogous to Sinkhorn balancing.

Decomposition We have no special-purpose tree decomposition algorithm, and therefore resort to a general decomposition algorithm for convex polytopes that is based on linear programming. Let w be in the tree polytope. Choose an arbitrary vertex C (i.e. a spanning tree) by minimising a linear objective over the current polytope. Now use linear programming to find the furthest point w' in the polytope on the ray from C through w. At least one more inequality constraint is tight for w'. Thus w' lies in a convex polytope of at least one dimension lower. Add this inequality constraint as an equality constraint, recursively decompose w', and express w as a convex combination of C and the decomposition of w'. The recursion bottoms out at a vertex (i.e. a spanning tree) and the total number of iterations is at most d.

Regret bound By (6.1), the regret $\mathbb{E}[\ell_{\mathrm{CH}}] - \ell^\star$ of CH on undirected and directed spanning trees is at most

$$\begin{aligned}\mathbb{E}[\ell_{\mathrm{CH}}] - \ell^\star &\leq \sqrt{2\ell^\star(n-1)\ln(n/2)} + (n-1)\ln(n/2),\\ \mathbb{E}[\ell_{\mathrm{CH}}] - \ell^\star &\leq \sqrt{2\ell^\star(n-1)\ln(n-1)} + (n-1)\ln(n-1).\end{aligned}$$

We provide matching lower bounds in Section 6.4.

6.4 Lower Bounds

Whereas it is easy to get some regret bounds with additional range factors, we show that CH is essentially optimal in all our applications. We leverage the following lower bound for the vanilla expert case:

6.4.1. Theorem. *There are positive constants c_1 and c_2 s.t. any online algorithm for q experts with loss range $[0, U]$ can be forced to have expected regret at least*

$$c_1\sqrt{\ell^\star U \ln q} + c_2 \ln q. \tag{6.8}$$

This type of bound was recently proven in [3]. Note that c_1 and c_2 are independent of the number of experts, the range of the losses and the algorithm. Earlier versions of the above lower bound using many quantifier and limit arguments are given in [28, 77]. We now prove lower bounds for our structured concept classes by embedding the original expert problem into each class and applying the above theorem. This type of reduction was pioneered in [77] for permutations.

The general reduction works as follows. We identify q structured concepts $C_1, \ldots, C_q$ in the concept class $\mathcal{C} \subseteq \{0,1\}^d$ to be learned that partition the d components. Now assume we have an online algorithm for learning class $\mathcal{C}$. From this we construct an algorithm for learning with q experts with loss range $[0, U]$. Let $\ell \in [0, U]^q$ denote the loss vector for the expert setting. From this we construct a loss vector $L \in [0,1]^d$ for learning $\mathcal{C}$: $L := \sum_{i=}^{q} \frac{\ell_i}{U} C_i$. That is, we spread the loss of expert i, evenly among the U many components used by concept C_i. Second, we transform the predictions as follows. Say our algorithm for learning $\mathcal{C}$ predicts with any structured concept $C \in \mathcal{C}$. Then we play expert i with probability $C_i \cdot C / U$. The expected loss of the expert algorithm now equals the transformed loss of the algorithm for learning concepts in $\mathcal{C}$:

$$\mathbb{E}[\ell_{\mathbf{i}}] \;=\; \sum_{i=1}^{q} \frac{C_i \cdot C}{U} \ell_i \;=\; C \cdot \sum_{i=1}^{q} \frac{\ell_i}{U} C_i \;=\; C \cdot L$$

This also means that the expected loss of the expert algorithm equals the expected loss of the algorithm for learning the structured class. This implies that the expected regret of the algorithm for learning $\mathcal{C}$ is at least the expected regret of the expert algorithm. The lower bound (6.8) for the regret in the expert setting is thus also a lower bound for the regret of the structured prediction task.

k-sets We assume that k divides n. Then we can partition $[d]$ with n/k sets, where set i uses components $(i-1)k+1, \ldots, ik$. The resulting lower bound has leading factor $\sqrt{k \ln \frac{n}{k}}$, matching the upper bound for CH within constant factors.

Truncated permutations We can partition the n^2 assignments into n full permutations. For example, the n cyclic shifts of the identity per-

mutation achieve this. The truncations to length k of those n permutations partition the kn components in the truncated case. The lower bound with leading factor $\sqrt{k \ln n}$ again matches the regret bound of CH within constant factors.

Spanning trees As observed in [72], the complete undirected graph has $(n-1)/2$ edge-disjoint spanning trees. Hence we get a lower bound with leading factor $\sqrt{(n-1)\ln((n-1)/2)}$. Each undirected spanning tree can be made directed by fixing a root. So there are at least as many disjoint directed spanning trees with a fixed root. In both cases we match the regret of CH within a constant factor.

Paths Consider the directed graph on $[n] \cup s, t$ that has n/k disjoint s-t paths of length $k+1$ connecting source to sink. By construction, we can embed n/k experts with loss range $[0, k]$ into this graph, so the regret has leading factor at least $\sqrt{k \log(n/k)}$. This graph is a subgraph of the complete directed graph $s \to K_n \to t$. Moreover, nature can force the algorithm to essentially play on the disjoint path graph by giving all edges outside it sheer infinite loss in a sheer infinite number of trials. This shows that the regret w.r.t. a comparator path of length k through the full graph has leading factor at least $\sqrt{k \log(n/k)}$.

A lower bound on the regret for arbitrary graphs is difficult to obtain since various interesting problems can be encoded as path problems. For example, the expert problem where each expert has a different loss range can be encoded into a graph that has a disjoint path of each length $1, 2, \ldots n$. The optimal algorithm for such expert problems was recently found in [2], but its regret has no closed form expression. It might be that the regret of CH is tight within constant factors for all graphs, but this question remains open.

6.5 Comparison to Other Algorithms

CH is a new member of an existing ecosystem. Other algorithms for structured prediction are EH[108] and FPL [92]. We now compare them.

Efficiency FPL can be readily applied efficiently to our examples of structured concept classes: k-sets take $O(n)$ per trial using variants

of median-finding, truncated permutations take $O(k^2 n)$ per trial using the Hungarian method for minimum weight bipartite matching, paths take $O(n^2)$ per trial using Dijkstra's shortest path algorithm and spanning trees take $O(n^2)$ per trial using either Prim's algorithm or Chu–Liu/Edmonds's algorithm for finding a minimum weight spanning tree.

EH can be efficiently implemented for k-sets [185] and paths [173] using dynamic programming, and for spanning trees [99] using the Matrix-Tree Theorem by Kirchoff (undirected) and Tutte (directed). An approximate implementation based on MCMC sampling could be built for permutations based upon [85].

In most cases FPL and EH are faster than CH. This may be partly due to the novelty of CH and the lack of special-purpose algorithms for it. On the other hand, FPL solves a linear minimisation problem, which is intuitively simpler than minimising a convex relative entropy.

6.5.1 Improved Regret Bounds with the Unit Rule

On the other hand, we saw in Section 6.2.2 that the general regret bound for CH (6.1) improves the guarantees of EH (6.1) by a factor $\sqrt{U}$ and those of FPL (6.3) by a larger factor $\sqrt{d}$. It is an open question whether these factors are real or simply an artifact of the bounding technique (see Section 6.6). We now give an example of a property of structured concept classes that makes these range factors vanish.

We say that a prediction algorithm has the *unit rule* on a given structured concept class $\mathcal{C}$ if its worst-case performance is achieved when in each trial only a single expert has nonzero loss. Without changing the prediction algorithm, the unit rule immediately improves its regret bound by reducing the effective loss range of each concept from $[0, U]$ to $[0, 1]$. The improved regret bounds are (c.f. (6.2) and (6.3))

$$\mathbb{E}\left[\ell_{\mathrm{EH}}\right] \leq \ell^\star + \sqrt{2\ell^\star \ln D} + \ln D \tag{6.9}$$

$$\mathbb{E}\left[\ell_{\mathrm{FPL}}\right] \leq \ell^\star + \sqrt{4\ell^\star U \ln d} + 3U \ln d \tag{6.10}$$

The unit rules for EH and FPL on experts have been observed before [92, 6]. We reprove them here for completeness. The unit rule holds for both EH and FPL on sets, and for EH on undirected trees. It fails for EH and FPL on permutations, and for EH on directed trees.

We prove the unit rule for EH on sets here, and counter it for EH on directed trees. All other proofs and counterexamples are delayed to Appendix 6.A.

6.5.1.1 Unit Rule Holds for EH on k-sets

Fix an expert i, and let j be an arbitrary other expert. We claim that if we hand out loss to i, then the usage of j increases. For each k-set S, we denote the prior weight of S by W_S. We abbreviate

$$\begin{aligned} Z_i &:= \sum_{S:i\in S} W_S, & Z_{\neg i} &:= \sum_{S:i\notin S} W_S, \\ Z_j &:= \sum_{S:j\in S} W_S, & Z_{\neg j} &:= \sum_{S:j\notin S} W_S, \\ Z_{i\wedge j} &:= \sum_{S:i\in S,j\in S} W_S, & Z_{\neg i\wedge j} &:= \sum_{S:i\notin S,j\in S} W_S, \\ Z_{i\wedge\neg j} &:= \sum_{S:i\in S,j\notin S} W_S, & Z_{\neg i\wedge\neg j} &:= \sum_{S:i\notin S,j\notin S} W_S. \end{aligned}$$

6.5.1. Theorem. *Assume that the prior weights have product structure, i.e.* $W_S \propto \prod_{i\in S} w_i$. *Then*

$$Z_j = \mathbb{P}(j \in \mathbf{S}^1) \;\le\; \mathbb{P}(j \in \mathbf{S}^2 | \ell^1 = \delta_i) = \frac{Z_{i\wedge j}\mathrm{e}^{-\eta} + Z_{\neg i\wedge j}}{Z_i\mathrm{e}^{-\eta} + Z_{\neg i}}.$$

Proof. With some rewriting, the claim is equivalent to

$$Z_i Z_j \;\ge\; Z_{i\wedge j} \qquad \text{and also} \qquad Z_{i\wedge\neg j} Z_{\neg i\wedge j} \;\ge\; Z_{i\wedge j} Z_{\neg i\wedge\neg j}$$

Define

$$R(n,k) := \sum_{\substack{S\subseteq[n]\\|S|=k}} \prod_{i\in S} w_i.$$

We now show that $R(n,k+1)R(n,m) \ge R(n,k)R(n,m+1)$ for all $0 \le k < m < n$. The proof proceeds by induction on n. The case $n = 0$ is trivial. Now suppose that the claim holds up to n. We need to show it for $n+1$. For $n > 0$, we have

$$R(n,k) \;=\; \mathbf{1}_{k>0} w_n R(n-1,k-1) + \mathbf{1}_{k<n} R(n-1,k). \tag{6.11}$$

Suppose that the induction hypothesis holds up to n. We must show that for all $0 \leq k < m < n+1$

$$R(n+1,k+1)R(n+1,m) \;\geq\; R(n+1,k)R(n+1,m+1).$$

By (6.11), this is equivalent to

$$\begin{aligned}&\big(w_{n+1}R(n,k) + \mathbf{1}_{k<n}R(n,k+1)\big)\big(\mathbf{1}_{m>0}w_{n+1}R(n,m-1) + \mathbf{1}_{m\leq n}R(n,m)\big) \;\geq \\ &\quad\big(\mathbf{1}_{k>0}w_{n+1}R(n,k-1) + \mathbf{1}_{k\leq n}R(n,k)\big)\big(\mathbf{1}_{m+1>0}w_{n+1}R(n,m) + \mathbf{1}_{m<n}R(n,m+1)\big).\end{aligned}$$

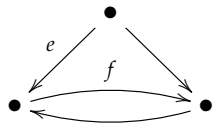
Now we expand, and use $0 \leq k < m < n+1$ to eliminate indicators. It remains to show

$$\begin{pmatrix} (w_{n+1})^2R(n,k)R(n,m-1) + \\ w_{n+1}R(n,k)R(n,m) + \\ w_{n+1}R(n,k+1)R(n,m-1) + \\ R(n,k+1)R(n,m)\end{pmatrix} \geq \begin{pmatrix} \mathbf{1}_{k>0}(w_{n+1})^2R(n,k-1)R(n,m) + \\ \mathbf{1}_{k>0}\mathbf{1}_{m<n}w_{n+1}R(n,k-1)R(n,m+1) + \\ w_{n+1}R(n,k)R(n,m) + \\ \mathbf{1}_{m<n}R(n,k)R(n,m+1)\end{pmatrix}$$

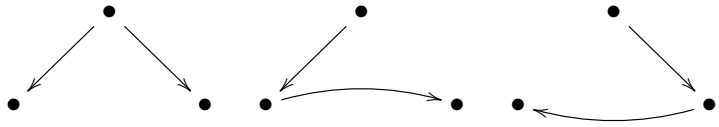
We now show that this inequality holds line-wise. Lines with active indicators trivially hold. If $k-1=m$, the second line holds with equality. Otherwise, and for the other lines we use the induction hypothesis. □

6.5.1.2 Unit Rule Fails for EH on Directed Spanning Trees

The unit rule is violated for EH on directed trees. Consider this graph



and its three directed spanning trees:



Note that we may always restrict attention to a given graph G by assigning zero prior weight to all spanning trees of the full graph that use edges outside G. Now if we put a unit of loss on edge e, the usage of f decreases, and vice versa, contradicting the unit rule. Call the prior

weights on directed trees W_A, W_B, W_C. Then the usages satisfy

$$W_A + W_B = \mathbb{P}(e \in \mathbf{T}^1) \geq \mathbb{P}(e \in \mathbf{T}^2 | \ell^1 = \delta_f) = \frac{W_A + W_B \mathrm{e}^{-\eta}}{W_A + W_B \mathrm{e}^{-\eta} + W_C},$$
$$W_B = \mathbb{P}(f \in \mathbf{T}^1) \geq \mathbb{P}(f \in \mathbf{T}^2 | \ell^1 = \delta_e) = \frac{W_B \mathrm{e}^{-\eta}}{W_A \mathrm{e}^{-\eta} + W_B \mathrm{e}^{-\eta} + W_C}.$$

Hence the unit rule is violated on directed spanning trees.

6.6 Conclusion

We developed the Component Hedge algorithm for online prediction over structured expert classes. The advantage of CH is that it has a general regret bound without the range factors that typically plague EH and FPL. We considered several example concept classes, and showed that the lower bound is matched in each case.

Open problems While the unit rule is one method for proving regret bounds for EH and FPL that are close to optimum, there might be other proof methods that show that EH and FPL perform as well as CH when applied to structured concepts. We know of no examples of structured concept classes where EH and FPL are clearly suboptimal. Resolving the question of whether such examples exist is our main open problem.

The prediction task for each structured concept class can be analysed as a two-player zero-sum game versus nature which tries to maximise the regret. The paper [6] gave an efficient implementation of the minimax optimal algorithm for playing against an adversary in the vanilla expert setting. Actually, the key insight was that the unit rule holds for the optimal algorithm in the vanilla expert case. This fact made it possible to design a balanced algorithm that incurs the same loss no matter which sequence of unit losses is chosen by nature. Unfortunately, the optimum algorithm does not satisfy the unit rule for any of the structured concept classes considered here. However, there might be some sort of relaxation of the unit rule that still leads to an efficient implementation of the optimum algorithm.

In this chapter the loss of a structured concept C always had the form $C \cdot \ell$, where ℓ is the loss vector for the components. This allowed us to maintain a mixture of concepts w and predict with a random

concept $\mathbf{C}$ s.t. $\mathbb{E}[\mathbf{C}] = w$. By linearity, the expected loss of such a randomly drawn concept $\mathbf{C}$ is the same as the loss of the mixture w. For regression problems with for example the convex loss $(C \cdot \ell - y)^2$ our algorithm can still maintain a mixture w, but now the expected loss of $\mathbf{C}$, i.e. $\mathbb{E}[(\mathbf{C} \cdot \ell - y)^2]$, is typically larger than the loss $(w \cdot \ell - y)^2$ of the mixture. We are confident that in this more general setting we can still get good regret bounds compared to the best mixture chosen in hind-sight. All we need to do is replace CH with the more general "Component Exponentiated Gradient" algorithm, which would do an EG update on the parameter vector w and project the updated vector back into the hull of the concepts.

In general, we believe that we have a versatile method of learning with structured concept classes. For example it is easy to augment the updates with a "share update" [80, 19] for the purpose of making them robust against sequences of examples where the best comparator changes over time. We also believe that our methods will get rid of the additional range factors in the bandit setting [26] and that gain versions of the algorithm CH also have good regret bounds.

At the core of our methods lies a relative entropy regularization which results in a multiplicative update on the components. In general, which relative entropy to choose is always one of the deepest questions. For example in the case of learning k-sets, a sum of binary relative entropies over the component can be used that incorporates the $w_i \leq 1$ constraints into the relative entropy term. In general incorporating inequality constraints into the relative entropy seems to have many advantages. However how to do this is an open ended research question.

6.A Unit rule

This appendix gives proofs of and counterexamples to the unit rule. We already saw some unit rule results in Section 6.5.1.

6.A.1 Unit Rule Holds for EH on Experts

Let $\mathbf{E}^1$ and $\mathbf{E}^2$ denote a random expert sampled by the algorithm in the first and second trial. Let δ_j denote a loss vector that assigns unit loss to expert j and zero loss to all other experts. Let W_i and W_j be the prior

weight of experts i and j. Then

$$W_j = \mathbb{P}(j = \mathbf{E}^1) \;\leq\; \mathbb{P}(j = \mathbf{E}^2 | \ell^1 = \delta_i) = \frac{W_j}{1 - W_i(1 - \mathrm{e}^{-\eta})}.$$

Thus, if we hand out loss to one expert, *all* other usages increase. This unit rule result does not lead to improved regret bounds, (6.2) and (6.9) already coincide for experts.

6.A.2 Unit Rule Fails for EH on Permutations

There are two permutations of size two: $A = \{(1 \leftarrow 1), (2 \leftarrow 2)\}$ and $B = \{(1 \leftarrow 2), (2 \leftarrow 1)\}$. To contradict the unit rule, we show that if we give a unit of loss to the component $(1 \leftarrow 1)$, then the usage of $(2 \leftarrow 2)$ goes down with it and vice versa. By symmetry, we only need to show it in one order. Let W_A, W_B denote the prior weights of the two permutations. Then the usages satisfy

$$W_A \;=\; \mathbb{P}\left((2 \leftarrow 2) \in \mathbf{\Pi}^1\right) \;\geq\; \mathbb{P}\left((2 \leftarrow 2) \in \mathbf{\Pi}^1 \middle| \ell^1 = \delta_{(1 \leftarrow 1)}\right) \;=\; \frac{W_A \mathrm{e}^{-\eta}}{W_A \mathrm{e}^{-\eta} + W_B},$$

where $\mathbb{P}$ denotes probability with respect to the algorithm's internal randomisation.

6.A.3 Unit Rule Holds for EH on Undirected Spanning Trees

Expanded Hedge on trees can be implemented using the Matrix-Tree Theorem by Kirchoff (undirected) and Tutte (directed). This was pioneered in [99] for log-loss. It can be easily adapted to dot loss. Sampling undirected spanning trees can be done using [23]. This method does not easily generalise to directed spanning trees. Computing the usages is fine for both, and this implies that computing the expected loss is fine for both as well. For directed spanning trees, we can first compute the usages and then decompose as for CH below.

The following theorem neatly characterises the log-partition function.

6.A.1. Theorem (Kirchhoff's Matrix-Tree Theorem). *Let G be an undirected graph, and let w assign weights to the edges of G. Let $\mathcal{S}$ be the set of*

spanning trees of G, and let L be the graph Laplacian of G, i.e. $L_{i,j} = w(i,j)$ *and* $L_{i,i} = -\sum_k w(i,k)$*. Then*

$$\sum_{T \in \mathcal{S}} \prod_{e \in T} w(e) \;=\; \det(L_{[1,1]}),$$

where $L_{[1,1]}$ *is the first minor of L, i.e. L excluding its first row and column.*

This theorem allows us to prove the unit rule for EH on undirected trees.

6.A.2. Theorem. *For all edges e, f*

$$Z \cdot Z_{\neg e \wedge \neg f} \;\leq\; Z_{\neg e} \cdot Z_{\neg f}$$

Simple case (*e* and *f* have a common vertex) The following theorem is essential.

6.A.3. Theorem. *For any numbers* a, b, c*, vectors* $\boldsymbol{v}, \boldsymbol{w}$ *and symmetric matrix R*

$$\det \begin{pmatrix} a & b & \boldsymbol{v}^T \\ b & c & \boldsymbol{w}^T \\ \boldsymbol{v} & \boldsymbol{w} & R \end{pmatrix} \det(R) \;\leq\; \det \begin{pmatrix} a & \boldsymbol{v}^T \\ \boldsymbol{v} & R \end{pmatrix} \det \begin{pmatrix} c & \boldsymbol{w}^T \\ \boldsymbol{w} & R \end{pmatrix}$$

Proof. First use the fact that $\det \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \det(D) \det(A - BD^{-1}C)$ repeatedly. We then need to show

$$\left(a - \begin{pmatrix} b & \boldsymbol{v}^T \end{pmatrix} \begin{pmatrix} c & \boldsymbol{w}^T \\ \boldsymbol{w} & R \end{pmatrix}^{-1} \begin{pmatrix} b \\ v^T \end{pmatrix} \right) (c - \boldsymbol{w}^T R \boldsymbol{w}) \det(R) \det(R) \;\leq$$
$$(a - \boldsymbol{v}^T R^{-1} \boldsymbol{v}) \det(R) \, (c - \boldsymbol{w}^T R \boldsymbol{w}) \det(R)$$

Now divide out $\det(R)^2$, which is positive, and use

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix}^{-1} = \begin{pmatrix} (A - BD^{-1}C)^{-1} & -(A - BD^{-1}C)^{-1} BD^{-1} \\ -D^{-1}C(A - BD^{-1}C)^{-1} & D^{-1} + D^{-1}B(A - BD^{-1}C)^{-1}CD^{-1} \end{pmatrix}$$

to obtain

$$\begin{pmatrix} b & \boldsymbol{v}^T \end{pmatrix} \begin{pmatrix} c & \boldsymbol{w}^T \\ \boldsymbol{w} & R \end{pmatrix}^{-1} \begin{pmatrix} b \\ v^T \end{pmatrix} \;=\; \frac{(b - \boldsymbol{v}^T R^{-1} \boldsymbol{w})^2}{c - \boldsymbol{w}^T R^{-1} \boldsymbol{w}} + \boldsymbol{v}^T R^{-1} \boldsymbol{v}.$$

It then remains to show

$$\left(a - \frac{(b - \boldsymbol{v}^T R^{-1} \boldsymbol{w})^2}{c - \boldsymbol{w}^T R^{-1} \boldsymbol{w}} - \boldsymbol{v}^T R^{-1} \boldsymbol{v}\right)(c - \boldsymbol{w}^T R \boldsymbol{w}) \;\leq\; (a - \boldsymbol{v}^T R^{-1} \boldsymbol{v})(c - \boldsymbol{w}^T R \boldsymbol{w})$$

which follows from

$$(b - \boldsymbol{v}^T R^{-1} \boldsymbol{w})^2 \;\geq\; 0. \qquad \square$$

Hard case (e and f have no common vertex)

6.A.4. Theorem. *Let a, b, c, d, e, f be numbers, $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}$ be vectors, and let R be a symmetric matrix. Then*

$$\begin{vmatrix} a & b & c & \boldsymbol{u}^T \\ b & d & e & \boldsymbol{v}^T \\ c & e & f & \boldsymbol{w}^T \\ \boldsymbol{u} & \boldsymbol{v} & \boldsymbol{w} & R \end{vmatrix} \left(\begin{vmatrix} d & \boldsymbol{v}^T \\ \boldsymbol{v} & R \end{vmatrix} + \begin{vmatrix} e & \boldsymbol{v}^T \\ \boldsymbol{w} & R \end{vmatrix} + \begin{vmatrix} e & \boldsymbol{w}^T \\ \boldsymbol{v} & R \end{vmatrix} + \begin{vmatrix} f & \boldsymbol{w}^T \\ \boldsymbol{w} & R \end{vmatrix} \right) \;\leq$$
$$\begin{vmatrix} d & e & \boldsymbol{v}^T \\ e & f & \boldsymbol{w}^T \\ \boldsymbol{v} & \boldsymbol{w} & R \end{vmatrix} \left(\begin{vmatrix} a & c & \boldsymbol{u}^T \\ c & f & \boldsymbol{w}^T \\ \boldsymbol{u} & \boldsymbol{w} & R \end{vmatrix} + \begin{vmatrix} a & b & \boldsymbol{u}^T \\ c & e & \boldsymbol{w}^T \\ \boldsymbol{u} & \boldsymbol{w} & R \end{vmatrix} + \begin{vmatrix} a & c & \boldsymbol{u}^T \\ b & e & \boldsymbol{v}^T \\ \boldsymbol{u} & \boldsymbol{w} & R \end{vmatrix} + \begin{vmatrix} a & b & \boldsymbol{u}^T \\ b & d & \boldsymbol{v}^T \\ \boldsymbol{u} & \boldsymbol{v} & R \end{vmatrix} \right)$$

Proof. Currently left to the reader :-). We need a good reduction from this case to the simple case. $\square$

6.A.4 Unit Rule Holds for FPL on Experts

The unit rule for FPL holds for all perturbations. Fix prior loss ℓ, and let $\boldsymbol{\rho}$ denote a random permutation vector. Then by monotonicity of probability distributions (the right set is bigger)

$$\mathbb{P}\left(\bigcap_{k \in [n]} \left\{ \ell_j + \tfrac{\rho}{\eta} \leq \ell_k + \tfrac{\rho}{\eta} \right\} \right) \;=\; \mathbb{P}(j = \mathbf{E}^1) \;\leq$$
$$\mathbb{P}(j = \mathbf{E}^2 | \ell^1 = \delta_i) \;=\; \mathbb{P}\left(\bigcap_{k \in [n]} \left\{ \ell_j + \tfrac{\rho}{\eta} \leq (\ell + \delta_i)_k + \tfrac{\rho}{\eta} \right\} \right).$$

By the unit rule, we may replace (6.3) by (6.10), eliminating a factor d from under the square root, yielding — up to constants — the same bound as EH. In fact, for Gumbel perturbations FPL coincides with EH. Unfortunately, this fact remains outside the scope of the general analysis of Section 6.2.2.

6.A.5 Unit Rule Holds for FPL on Sets

The usage of component i equals the probability that we draw i when we draw k items without replacement, with probabilities proportional to their weight. Let $0 \leq \beta \leq 1$. Define

$$w_S := \sum_{h \in S} w_h, \qquad R(i, S, 0) := 0, \qquad R(i, j, S, 0) := 0,$$

and recursively

$$\begin{aligned} R(i, S, k) &:= \frac{w_i + \sum_{h \in S} w_h R(i, S - h, k-1)}{w_i + w_S} \\ R(i, j, S, k) &:= \frac{w_i + w_j R(i, S, k-1) + \sum_{h \in S} w_h R(i, S - h, k-1)}{w_i + w_j + w_S} \\ \tilde{R}(i, j, S, k) &:= \frac{w_i + \beta w_j R(i, S, k-1) + \sum_{h \in S} w_h \tilde{R}(i, S - h, k-1)}{w_i + \beta w_j + w_S} \end{aligned}$$

6.A.5. THEOREM. *For all* $0 \leq k \leq n = |S| + 2$.

$$R(i, j, S, k) = \mathbb{P}(i \in \mathbf{S}^1) \;\leq\; \mathbb{P}(i \in \mathbf{S}^2 | \ell^1 = \delta_j) = \tilde{R}(i, j, S, k).$$

Proof. By induction on k. Equality holds for $k = 0$. Suppose the theorem holds up till k. We need to show

$$\frac{w_i + w_j R(i, S, k) + \sum_{h \in S} w_h R(i, S - h, k)}{w_i + w_j + w_S} \leq \frac{w_i + \beta w_j R(i, S, k) + \sum_{h \in S} w_h \tilde{R}(i, S - h, k)}{w_i + \beta w_j + w_S}$$

We apply the induction hypothesis, multiply by both denominators, rearrange and divide by $(1 - \beta) w_j (w_i + w_S)$. It then suffices to show

$$R(i, S, k+1) = \frac{w_i + \sum_{h \in S} w_h R(i, S - h, k)}{w_i + w_S} \geq R(i, S, k)$$

which is obvious. □

6.A.6 Unit Rule Fails for FPL on Permutations

The perturbed loss of a permutation is the loss of that perturbation plus the sum of *two* independent perturbations. Initially both permutations have loss zero, so that either permutation is the perturbed leader with probability one half. If component $(1 \leftarrow 1)$ suffers loss, then obviously the usage of $(2 \leftarrow 2)$ goes down.

6.B Dual Problems for $\triangle$-projection

In this appendix, we compute the Lagrange dual problems to the relative entropy projections on (truncated) permutations, paths and spanning trees. The dual problems involve less variables and less constraints, and thus lead to more efficient implementation of the projection.

6.B.1 Matching Polytope

We want to find

$$\operatorname*{argmin}_{w \text{ s.t. } (6.5)} \triangle(w\|\widehat{w}).$$

Introduce Lagrange multipliers λ_i and μ_j for each row and column constraint. Form the Lagrangian

$$F(w,\lambda,\mu) = \triangle(w\|\widehat{w}) + \sum_{i\in[k]} \lambda_i \Bigg(\sum_{j\in[n]} w_{i\leftarrow j} - 1 \Bigg) + \sum_{j\in[n]} \mu_j \Bigg(\sum_{i\in[k]} w_{i\leftarrow j} - 1 \Bigg)$$

Equating the derivative of F w.r.t. w to zero yields $w^{i\leftarrow j} = \widehat{w}_{i\leftarrow j} \mathrm{e}^{-\lambda_i - \mu_j}$. So the dual function is

$$F(\lambda,\mu) := \inf_{w\in\mathbb{R}^d} F(w,\lambda,\mu) = \sum_{i\in[k]} \sum_{j\in[n]} (1 - \mathrm{e}^{-\lambda_i-\mu_j}) \widehat{w}_{i\leftarrow j} - \sum_{i\in[k]} \lambda_i - \sum_{j\in[n]} \mu_j$$

The advantage of the dual problem is that we only have $k+n$ variables, whereas the primal has kn. But since μ correspond to inequality constraints, we have to maximise the dual function F under the constraint $\mu \geq \mathbf{0}$.

6.B.2 Flow Polytope

We want to find

$$\operatorname*{argmin}_{w \text{ a flow (6.6)}} \triangle(w\|\widehat{w}).$$

Here we generalise slightly. We allow flow from a node to itself. And we allow flow to enter the source. It is still forbidden for flow to leave the sink, and the source still has unit excess flow. We introduce a Lagrange multiplier λ_i, for each node/constraint $i \neq t$, and form the Lagrangian

$$F(w,\lambda) = \triangle(w\|\widehat{w}) + \lambda_s + \sum_{i\neq t}\lambda_i\left(\sum_{j\neq t} w_{j,i} - \sum_j w_{i,j}\right)$$

Equating the derivative of F w.r.t. w to zero yields $w_{i,j} = \widehat{w}_{i,j}\mathrm{e}^{\lambda_i-\lambda_j}$, with the convention that $\lambda_t = 0$. The concave dual function thus equals

$$F(\lambda) := \inf_{w\in\mathbb{R}^d} F(w,\lambda) = \lambda_s + \sum_{i\neq t}\sum_j (1-\mathrm{e}^{\lambda_i-\lambda_j})\widehat{w}_{i,j}.$$

The advantage of the dual problem is that we now only have $n+1$ variables and no constraints. The primal problem has order n^2 variables and $n+1$ equality constraints. By strong duality, the optimal primal variables w^* can be reconstructed from the optimum dual variables λ^*.

6.B.3 Tree Polytope

Setup Fix any $\widehat{w} \in \mathbb{R}^d_+$. We are interested in finding

$$\operatorname*{argmin}_{w \text{ s.t. (6.7)}} \triangle(w\|\widehat{w}).$$

We introduce Lagrange multipliers α^k_{ij}, γ^k_{ij}, λ and μ^k_i, and form the Lagrangian

$$\begin{aligned} F(w,\alpha,\gamma,\lambda,\mu) \;:=\; & \triangle(w\|\widehat{w}) - \sum_{i,j\neq i,k\neq 1}\alpha^k_{ij}f^k_{ij} + \sum_{i,j\neq i,k\neq 1}\gamma^k_{ij}\left(f^k_{ij}-w_{ij}\right) \\ & + \lambda\left(\sum_{i,j\neq i} w_{ij} - (n-1)\right) + \sum_{i,k\neq 1}\mu^k_i\left(\sum_{j\neq i}\left(f^k_{ij}-f^k_{ji}\right)+\theta^k_i\right). \end{aligned}$$

Lagrange dual function The partial derivatives are

$$\frac{\partial F}{\partial w_{ij}} = \log \frac{w_{ij}}{\widehat{w}_{ij}} + \lambda - \sum_{k\neq 1} \gamma_{ij}^k, \qquad \frac{\partial F}{\partial f_{ij}^k} = \mu_i^k - \mu_j^k + \gamma_{ij}^k - \alpha_{ij}^k.$$

By setting the partial derivatives to zero, we obtain the Lagrange dual

$$\begin{aligned} F^* := \inf_{w_{ij}, f_{ij}^k} F &= \sum_{i,j\neq i} w_{ij}^0 - \mathrm{e}^{-\lambda} \sum_{i,j\neq i} \widehat{w}_{ij} \mathrm{e}^{\sum_{k\neq 1} \gamma_{ij}^k} - \lambda\,(n-1) + \sum_{i,k\neq 1} \mu_i^k \theta_i^k \\ &= \sum_{i,j\neq i} \widehat{w}_{ij} - \mathrm{e}^{-\lambda} \sum_{i,j\neq i} \widehat{w}_{ij} \mathrm{e}^{\sum_{k\neq 1} \gamma_{ij}^k} - \lambda\,(n-1) + \sum_{k\neq 1} \left(\mu_k^k - \mu_1^k\right) \end{aligned}$$

Lagrange dual problem We now maximise the Lagrange dual F^* over the dual variables α_{ij}^k, γ_{ij}^k, μ_i^k and λ subject to the constraints

$$\alpha_{ij}^k \geq 0, \qquad \gamma_{ij}^k \geq 0, \qquad \mu_i^k - \mu_j^k + \gamma_{ij}^k - \alpha_{ij}^k = 0.$$

Since α_{ij}^k do not appear in the dual, these are equivalent to

$$\gamma_{ij}^k \geq 0, \qquad \gamma_{ij}^k \geq \mu_j^k - \mu_i^k.$$

Eliminating λ Note that λ is unconstrained. Its derivative is

$$\frac{\partial F^*}{\partial \lambda} = \mathrm{e}^{-\lambda} \sum_{i,j\neq i} \widehat{w}_{ij} \mathrm{e}^{\sum_{k\neq 1} \gamma_{ij}^k} - (n-1)\,.$$

Setting the derivative to zero, we obtain

$$\begin{aligned} F^\circ &:= \sup_{\lambda} F^* = \\ &\sum_{i,j\neq i} \widehat{w}_{ij} - (n-1) - (n-1) \ln \frac{\sum_{i,j\neq i} \widehat{w}_{ij} \mathrm{e}^{\sum_{k\neq 1} \gamma_{ij}^k}}{n-1} + \sum_{k\neq 1} \left(\mu_k^k - \mu_1^k\right) \end{aligned}$$

Eliminating γ_{ij}^k Since F° is decreasing in γ_{ij}^k, they each have to be set to their lower bound $\max\{0, \mu_j^k - \mu_i^k\}$. We get

$$\begin{aligned} F^\dagger := \sup_{\gamma_{ij}^k} F^\circ &= \sum_{i,j\neq i} \widehat{w}_{ij} - (n-1) - \\ &\qquad (n-1) \ln \frac{\sum_{i,j\neq i} \widehat{w}_{ij} \mathrm{e}^{\sum_{k\neq 1} \max\{0, \mu_j^k - \mu_i^k\}}}{n-1} + \sum_{k\neq 1} \left(\mu_k^k - \mu_1^k\right). \end{aligned}$$

Recovering primal variables Say that we have μ_i^k. We now want to extract the primal variables w_{ij} and f_{ij}^k from these. First we solve for all other dual variables in terms of μ_i^k:

$$\begin{aligned}
\gamma_{ij}^k &= \max\{0, \mu_j^k - \mu_i^k\} \\
\alpha_{ij}^k &= \mu_i^k - \mu_j^k + \gamma_{ij}^k = \max\{0, \mu_i^k - \mu_j^k\} = \gamma_{ji}^k \\
\lambda &= \ln \frac{\sum_{i,j\neq i} \widehat{w}_{ij} \mathrm{e}^{\sum_{k\neq 1} \gamma_{ij}^k}}{n-1} = \ln \frac{\sum_{i,j\neq i} \widehat{w}_{ij} \mathrm{e}^{\sum_{k\neq 1} \max\{0, \mu_j^k - \mu_i^k\}}}{n-1}
\end{aligned}$$

We then solve for the primal variables using the KKT conditions

$$\begin{aligned}
w_{ij} &= \widehat{w}_{ij} \mathrm{e}^{-\lambda + \sum_{k\neq 1} \gamma_{ij}^k} = \frac{(n-1)\, \widehat{w}_{ij} \mathrm{e}^{\sum_{k\neq 1} \max\{0, \mu_j^k - \mu_i^k\}}}{\sum_{i,j\neq i} \widehat{w}_{ij} \mathrm{e}^{\sum_{k\neq 1} \max\{0, \mu_j^k - \mu_i^k\}}} \\
f_{ij}^k &= \begin{cases} w_{ij} & \mu_j^k > \mu_i^k, \text{ i.e. } \gamma_{ij}^k > 0 \\ 0 & \mu_j^k < \mu_i^k, \text{ i.e. } \alpha_{ij}^k > 0 \end{cases}
\end{aligned}$$

Bibliography

[1] N. Abe and P. M. Long. Associative reinforcement learning using linear probabilistic concepts. In *Proc. 16th International Conf. on Machine Learning*, pages 3–11. Morgan Kaufmann, San Francisco, CA, 1999.

[2] J. Abernethy and M. K. Warmuth. Repeated games against budgeted adversaries. Unpublished manuscript.

[3] J. Abernethy, M. K. Warmuth, and J. Yellin. When random play is optimal against an adversary. Journal version of [6], in progress.

[4] J. Abernethy, J. Langford, and M. K. Warmuth. Continuous experts and the binning algorithm. In G. Lugosi and H. Simon, editors, *Learning Theory*, volume 4005 of *Lecture Notes in Computer Science*, pages 544–558. Springer Berlin / Heidelberg, 2006.

[5] J. Abernethy, E. Hazan, and A. Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In *In Proceedings of the 21st Annual Conference on Learning Theory (COLT*, 2008.

[6] J. Abernethy, M. K. Warmuth, and J. Yellin. Optimal strategies for random walks. In *Proceedings of The 21st Annual Conference on Learning Theory*, pages 437–446, July 2008.

[7] R. Ananthanarayanan, S. K. Esser, H. D. Simon, and D. S. Modha. The cat is out of the bag: cortical simulations with 10^9 neurons,

10^{13} synapses. In *SC '09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, pages 1–12, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-744-8.

[8] M. Anthony and N. Biggs. *Computational Learning Theory*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, UK, 1992.

[9] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda. Purposive behavior acquisition for a real robot by vision-based reinforcement learning. *Machine Learning*, 23:279–303, 1996.

[10] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2/3):235–256, 2002.

[11] L. Baird. Residual algorithms: reinforcement learning with function approximation. In *Proc. 12th International Conference on Machine Learning*, pages 30–37. Morgan Kaufmann, 1995.

[12] P. L. Bartlett and J. Baxter. Estimation and approximation bounds for gradient-based reinforcement learning. In *Proc. 13th Annu. Conference on Comput. Learning Theory*, pages 133–141. Morgan Kaufmann, San Francisco, 2000.

[13] P. L. Bartlett and J. Baxter. Estimation and approximation bounds for gradient-based reinforcement learning. *J. Comput. Syst. Sci.*, 64(1):133–150, 2002. Special Issue for COLT 2000.

[14] J. Baxter and P. L. Bartlett. Reinforcement learning in POMDP's via direct gradient ascent. In *Proc. 17th International Conf. on Machine Learning*, pages 41–48. Morgan Kaufmann, San Francisco, CA, 2000.

[15] J. O. Berger. Could Fisher, Jeffreys and Neyman have agreed on testing? *Statistical Science*, 18(1):1–32, 2003.

[16] K. Binmore. *Fun and games: a text on game theory*. D.C. Heath, 1991.

[17] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387310738.

[18] O. Bousquet. A note on parameter tuning for on-line shifting algorithms. Technical report, Max Planck Institute for Biological Cybernetics, 2003.

[19] O. Bousquet and M. K. Warmuth. Tracking a small set of experts by mixing past posteriors. *Journal of Machine Learning Research*, 3: 363–396, 2002.

[20] M. Bowling. Convergence problems of general-sum multiagent reinforcement learning. In *Proc. 17th International Conf. on Machine Learning*, pages 89–94. Morgan Kaufmann, San Francisco, CA, 2000.

[21] R. Bracewell. "Convolution" and "Two-dimensional convolution". In *The Fourier Transform and Its Applications*, pages 25–50 and 243–244. McGraw-Hill, 1965.

[22] R. I. Brafman and M. Tennenholtz. R-MAX - A general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2002.

[23] A. Broder. Generating random spanning trees. In *SFCS '89: Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 442–447, Washington, DC, USA, 1989. IEEE Computer Society. ISBN 0-8186-1982-1.

[24] N. Cesa-Bianchi and P. Fischer. Finite-time regret bounds for the multiarmed bandit problem. In *Proc. 15th International Conf. on Machine Learning*, pages 100–108. Morgan Kaufmann, San Francisco, CA, 1998.

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

[26] N. Cesa-Bianchi and G. Lugosi. Combinatorial bandits. In *Proceedings of the 22nd Annual Conference on Learning Theory*, 2009.

[27] N. Cesa-Bianchi, Y. Freund, D. P. Helmbold, and M. K. Warmuth. On-line prediction and conversion strategies. *Machine Learning*, 25:71–110, 1996. An extended abstract appeared in *EuroColt '93*.

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

[29] G.-H. Chen, M.-Y. Kao, Y.-D. Lyuu, and H.-K. Wong. Optimal buy-and-hold strategies for financial markets with bounded daily returns. In *Proc. of the 31st annual ACM symposium on Theory of computing*, pages 119–128. ACM, 1999. ISBN 1-58113-067-8.

[30] P. Cichosz and J. J. Mulawka. Fast and efficient reinforcement learning with truncated temporal differences. In *Proc. 12th International Conference on Machine Learning*, pages 99–107. Morgan Kaufmann, 1995.

[31] J. G. Cleary, Ian, and I. H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32:396–402, 1984.

[32] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19 (90):297–301, 1965. ISSN 00255718.

[33] T. M. Cover. Universal portfolios. *Mathematical Finance*, 1(1):1–29, 1991.

[34] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 1991.

[35] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK, 2000.

[36] R. H. Crites and A. G. Barto. Elevator group control using multiple reinforcement learning agents. *Machine Learning*, 33(2/3): 235–262, 1998.

[37] F. A. Dahl. A reinforcement learning algorithm applied to simplified two-player texas hold'em poker. In *Machine Learning: ECML 2001, 12th European Conference on Machine Learning, Freiburg, Germany, September 5-7, 2001, Proceedings*, volume 2167 of *Lecture Notes in Artificial Intelligence*, pages 85–96. Springer, 2001.

[38] E. Dannoura and K. Sakurai. An improvement on El-Yaniv-Fiat-Karp-Turpin's money-making bi-directional trading strategy. *IPL*, 66(1):27–33, 1998.

[39] A. P. Dawid. Statistical theory: The prequential approach. *Journal of the Royal Statistical Society, Series A*, 147, Part 2:278–292, 1984.

[40] A. P. Dawid, S. de Rooij, G. Shafer, A. Shen, N. Vereshchagin, and V. Vovk. Insuring against loss of evidence in game-theoretic probability. *Statistics & Probability Letters*, 81(1):157 – 162, 2011. ISSN 0167-7152.

[41] S. de Rooij and T. van Erven. Learning the switching rate by discretising Bernoulli sources online. In *JMLR Workshop and Conference Proceedings*, volume 5: AISTATS, 2009.

[42] G. DeJong. Hidden strengths and limitations: An empirical investigation of reinforcement learning. In *Proc. 17th International Conf. on Machine Learning*, pages 215–222. Morgan Kaufmann, San Francisco, CA, 2000.

[43] P. DeMarzo, I. Kremer, and Y. Mansour. Online trading algorithms and robust option pricing. In *Proc. of the 38 annual ACM symposium on Theory of computing*, pages 477–486. ACM, 2006. ISBN 1-59593-134-1.

[44] T. G. Dietterich. The MAXQ method for hierarchical reinforcement learning. In *Proc. 15th International Conf. on Machine Learning*, pages 118–126. Morgan Kaufmann, San Francisco, CA, 1998.

[45] T. G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

[46] T. G. Dietterich and N. S. Flann. Explanation-based learning and reinforcement learning: a unified view. *Machine Learning*, 28:169–210, 1997. Earlier version in 12th International Conf on ML, 1995.

[47] T. G. Dietterich and X. Wang. Support vectors for reinforcement learning. In *Machine Learning: ECML 2001, 12th European Conference on Machine Learning, Freiburg, Germany, September 5-7, 2001,*

Proceedings, volume 2167 of *Lecture Notes in Artificial Intelligence*, page 600. Springer, 2001.

[48] C. Domingo. Faster near-optimal reinforcement learning: Adding adaptiveness to the E^3 algorithm. In *Algorithmic Learning Theory, 10th International Conference, ALT '99, Tokyo, Japan, December 1999, Proceedings*, volume 1720 of *Lecture Notes in Artificial Intelligence*, pages 241–251. Springer, 1999.

[49] K. Driessens, J. Ramon, and H. Blockeel. Speeding up relational reinforcement learning through the use of an incremental first order decision tree learner. In *Machine Learning: ECML 2001, 12th European Conference on Machine Learning, Freiburg, Germany, September 5-7, 2001, Proceedings*, volume 2167 of *Lecture Notes in Artificial Intelligence*, pages 97–108. Springer, 2001.

[50] M. O. Duff. Q-learning for bandit problems. In *Proc. 12th International Conference on Machine Learning*, pages 209–217. Morgan Kaufmann, 1995.

[51] S. Džeroski, L. De Raedt, and H. Blockeel. Relational reinforcement learning. In *Proc. 15th International Conf. on Machine Learning*, pages 136–143. Morgan Kaufmann, San Francisco, CA, 1998.

[52] S. Dzeroski, L. De Raedt, and K. Driessens. Relational reinforcement learning. *Machine Learning*, 43(1/2):7–52, 2001.

[53] R. El-Yaniv, A. Fiat, R. M. Karp, and G. Turpin. Optimal search and one-way trading online algorithms. *Algorithmica*, 30(1):101–139, 2001.

[54] D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6: 503–556, 2005.

[55] E. Even-Dar, S. Mannor, and Y. Mansour. PAC bounds for multi-armed bandit and Markov decision processes. In *15th Annual Conference on Computational Learning Theory, COLT 2002, Sydney, Australia, July 2002, Proceedings*, volume 2375 of *Lecture Notes in Artificial Intelligence*, pages 255–270. Springer, 2002.

[56] C. N. Fiechter. Efficient reinforcement learning. In *Proc. 7th Annu. ACM Conf. on Comput. Learning Theory*, pages 88–97. ACM Press, New York, NY, 1994.

[57] D. J. Finton and Y. H. Hu. Importance-based feature extraction for reinforcement learning. In T. Petsche, editor, *Computational Learning Theory and Natural Learning Systems*, volume III: Selecting Good Models, chapter 5, pages 77–94. MIT Press, 1995.

[58] J. S. Frame. Mean deviation of the binomial distribution. *The American Mathematical Monthly*, 52(7):377–379, 1945.

[59] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.

[60] Z. Gábor, Z. Kalmár, and C. Szepesvári. Multi-criteria reinforcement learning. In *Proc. 15th International Conf. on Machine Learning*, pages 197–205. Morgan Kaufmann, San Francisco, CA, 1998.

[61] L. M. Gambardella and M. Dorigo. Ant-Q:a reinforcement learning approach to the traveling salesman problem. In *Proc. 12th International Conference on Machine Learning*, pages 252–260. Morgan Kaufmann, 1995.

[62] F. Garcia and S. M. Ndiaye. A learning rate analysis of reinforcement learning algorithms in finite-horizon. In *Proc. 15th International Conf. on Machine Learning*, pages 215–223. Morgan Kaufmann, San Francisco, CA, 1998.

[63] P. Geibel. Reinforcement learning with bounded risk. In *Proc. 18th International Conf. on Machine Learning*, pages 162–169. Morgan Kaufmann, San Francisco, CA, 2001.

[64] S. Geulen, B. Voecking, and M. Winkler. Regret minimization for online buffering problems using the weighted majority algorithm. In A. T. Kalai and M. Mohri, editors, *Proceedings of the 23rd Conference on Learning Theory*, pages 132–143. Omnipress, June 2010.

[65] Z. Ghahramani and G. E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12(4):831–864, 2000.

[66] Z. Ghahramani and M. I. Jordan. Factorial hidden markov models. *Machine Learning*, 29(2-3):245–273, 1997. ISSN 0885-6125.

[67] M. Ghavamzadeh and S. Mahadevan. Continuous-time hierarchial reinforcement learning. In *Proc. 18th International Conf. on Machine Learning*, pages 186–193. Morgan Kaufmann, San Francisco, CA, 2001.

[68] M. R. Glickman and K. Sycara. Evolutionary search, stochastic policies with memory, and reinforcement learning with hidden state. In *Proc. 18th International Conf. on Machine Learning*, pages 194–201. Morgan Kaufmann, San Francisco, CA, 2001.

[69] D. E. Goldberg. Probability matching, the magnitude of reinforcement, and classifier system bidding. *Machine Learning*, 5:407–425, 1990.

[70] R. B. Gramacy, M. K. Warmuth, S. A. Brandt, and I. Ari. Adaptive caching by refetching. In *In Advances in Neural Information Processing Systems 15*, pages 1465–1472. MIT Press, 2002.

[71] P. D. Grünwald. *The Minimum Description Length Principle*. The MIT Press, 2007.

[72] D. Gusfield. Connectivity and edge-disjoint spanning trees. *Information Processing Letters*, 16(2):87–89, 1983.

[73] J. Hannan. Approximation to Bayes risk in repeated play. In *Contributions to the Theory of Games*, volume 3, pages 97–139. Princeton University Press, 1957.

[74] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer Verlag, 2001.

[75] D. Haussler, J. Kivinen, and M. K. Warmuth. Sequential prediction of individual sequences under general loss functions. *IEEE Transactions on Information Theory*, 44(5):1906–1925, 1998.

[76] M. Heger. Consideration of risk in reinforcement learning. In *Proc. 11th International Conference on Machine Learning*, pages 105–111. Morgan Kaufmann, 1994.

[77] D. P. Helmbold and M. K. Warmuth. Learning permutations with exponential weights. *Journal of Machine Learning Research*, 10:1705–1736, July 2009.

[78] D. P. Helmbold, D. D. E. Long, T. L. Sconyers, and B. Sherrod. Adaptive disk spin-down for mobile computers. *ACM/Baltzer Mobile Networks and Applications (MONET)*, pages 285–297, 2000.

[79] M. Herbster and M. K. Warmuth. Tracking the best expert. In *Proceedings of the 12th Annual Conference on Learning Theory (COLT 1995)*, pages 286–294, 1995.

[80] M. Herbster and M. K. Warmuth. Tracking the best expert. *Machine Learning*, 32:151–178, 1998.

[81] M. Herbster and M. K. Warmuth. Tracking the best linear predictor. *Journal of Machine Learning Research*, 1:281–309, 2001.

[82] D. F. Hougen, M. Gini, and J. Slagle. An integrated connectionist approach to reinforcement learning for robotic control: The advantages of indexed partitioning. In *Proc. 17th International Conf. on Machine Learning*, pages 383–390. Morgan Kaufmann, San Francisco, CA, 2000.

[83] J. Hu and M. P. Wellman. Multiagent reinforcement learning: theoretical framework and an algorithm. In *Proc. 15th International Conf. on Machine Learning*, pages 242–250. Morgan Kaufmann, San Francisco, CA, 1998.

[84] M. Hutter and J. Poland. Adaptive online prediction by following the perturbed leader. *Journal of Machine Learning Research*, 6:639–660, Apr. 2005.

[85] M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM*, 51(4):671–697, 2004. ISSN 0004-5411.

[86] L. ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8:293–321, 1992.

[87] V. Kachitvichyanukul and B. W. Schmeiser. Binomial random variate generation. *Commun. ACM*, 31(2):216–222, 1988. ISSN 0001-0782.

[88] L. P. Kaelbling. Associative methods in reinforcement learning: an empirical study. In S. J. Hanson, T. Petsche, R. L. Rivest, and M. Kearns, editors, *Computational Learning Theory and Natural Learning Systems*, volume II: Intersections Between Theory and Experiment, chapter 9, pages 133–153. MIT Press, 1994.

[89] L. P. Kaelbling. Associative reinforcement learning: Functions in *k*-DNF. *Machine Learning*, 15(3):279–298, 1994.

[90] L. P. Kaelbling. Associative reinforcement learning: A generate and test algorithm. *Machine Learning*, 15(3):299–319, 1994.

[91] A. Kalai. A perturbation that makes "Follow the Leader" equivalent to "Randomized Weighted Majority". Private communication, Dec. 2005.

[92] A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005. ISSN 0022-0000.

[93] M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. In *Proc. 15th International Conf. on Machine Learning*, pages 260–268. Morgan Kaufmann, San Francisco, CA, 1998.

[94] M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002.

[95] M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, Massachusetts, 1994.

[96] H. Kimura and S. Kobayashi. An analysis of actor/critic algorithms using eligibility traces: reinforcement learning with imperfect value functions. In *Proc. 15th International Conf. on Machine Learning*, pages 278–286. Morgan Kaufmann, San Francisco, CA, 1998.

[97] H. Kimura, M. Yamamura, and S. Kobayashi. Reinforcement learning by stochastic hill climbing on discounted reward. In *Proc. 12th International Conference on Machine Learning*, pages 295–303. Morgan Kaufmann, 1995.

[98] H. Kimura, K. Miyazaki, and S. Kobayashi. Reinforcement learning in POMDPs with function approximation. In *Proc. 14th International Conference on Machine Learning*, pages 152–160. Morgan Kaufmann, 1997.

[99] T. Koo, A. Globerson, X. Carreras, and M. Collins. Structured prediction models via the Matrix-Tree theorem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150, 2007.

[100] W. M. Koolen and S. de Rooij. Combining expert advice efficiently. In R. Servedio and T. Zang, editors, *Proceedings of the 21st Annual Conference on Learning Theory (COLT 2008)*, pages 275–286, June 2008.

[101] W. M. Koolen and S. de Rooij. Combining expert advice efficiently. arXiv:0802.2015, Feb. 2008.

[102] D. Kuzmin and M. K. Warmuth. Optimum follow the leader algorithm. In *Proceedings of the 18th Annual Conference on Learning Theory (COLT '05)*, pages 684–686. Springer-Verlag, June 2005. Open problem.

[103] M. G. Lagoudakis and M. L. Littman. Algorithm selection using reinforcement learning. In *Proc. 17th International Conf. on Machine Learning*, pages 511–518. Morgan Kaufmann, San Francisco, CA, 2000.

[104] N. Landwehr. Modeling interleaved hidden processes. In *Proceedings of the 25th international conference on Machine learning*, pages 520–527, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4.

[105] M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proc.*

17th International Conf. on Machine Learning, pages 535–542. Morgan Kaufmann, San Francisco, CA, 2000.

[106] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag New York, Inc., 1997.

[107] L. Lin. Self-improving reactive agents: case studies of reinforcement learning frameworks. Technical Report CMU-CS-90-109, Carnegie Mellon Computer Science Department, Aug. 1990.

[108] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994. Preliminary version appeared in the Proceedings of the 30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, 1989.

[109] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proc. 11th International Conference on Machine Learning*, pages 157–163. Morgan Kaufmann, 1994.

[110] M. L. Littman and C. Szepesvári. A generalized reinforcement-learning model: Convergence and applications. In *Proc. 13th International Conference on Machine Learning*, pages 310–318. Morgan Kaufmann, 1996.

[111] R. Maclin and J. W. Shavlik. Creating advice-taking reinforcement learners. *Machine Learning*, 22:251–281, 1996.

[112] T. L. Magnanti and L. A. Wolsey. Optimal trees. In M. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 503–615. North-Holland, 1995.

[113] S. Mahadevan. To discount or not to discount in reinforcement learning: a case study comparing R learning and Q learning. In *Proc. 11th International Conference on Machine Learning*, pages 164–172. Morgan Kaufmann, 1994.

[114] S. Mahadevan. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning*, 22: 159–195, 1996.

[115] S. Mahadevan. Sensitive discount optimality: unifying discounted and average reward reinforcement learning. In *Proc. 13th International Conference on Machine Learning*, pages 328–336. Morgan Kaufmann, 1996.

[116] S. Mahadevan and J. Connell. Automatic programming of behavior-based robots using reinforcement learning. Technical report, IBM Research at Yorktown Heights, Dec. 1990.

[117] S. Mannor and N. Shimkin. A geometric approach to multi-criterion reinforcement learning. *Journal of Machine Learning Research*, 5:325–360, 2004.

[118] S. Mannor and J. N. Tsitsiklis. The sample complexity of exploration in the multi-armed bandit problem. *Journal of Machine Learning Research*, 5:623–648, 2004.

[119] Y. Mansour. Reinforcement learning and mistake bounded algorithms. In *Proc. 12th Annu. Conf. on Comput. Learning Theory*, pages 183–192. ACM Press, New York, NY, 1999.

[120] C. E. Mariano and E. F. Morales. DQL: A new updating strategy for reinforcement learning based on Q-learning. In *Machine Learning: ECML 2001, 12th European Conference on Machine Learning, Freiburg, Germany, September 5-7, 2001, Proceedings*, volume 2167 of *Lecture Notes in Artificial Intelligence*, pages 324–335. Springer, 2001.

[121] R. A. McCallum. Instance-based utile distinctions for reinforcement learning with hidden state. In *Proc. 12th International Conference on Machine Learning*, pages 387–395. Morgan Kaufmann, 1995.

[122] A. McGovern and A. G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proc. 18th International Conf. on Machine Learning*, pages 361–368. Morgan Kaufmann, San Francisco, CA, 2001.

[123] A. McGovern, E. Moss, and A. G. Barto. Building a basic block instruction scheduler with reinforcement learning and rollouts. *Machine Learning*, 49(2-3):141–160, 2002.

[124] O. Mihatsch and R. Neuneier. Risk-sensitive reinforcement learning. *Machine Learning*, 49(2-3):267–290, 2002.

[125] J. D. R. Millán and C. Torras. A reinforcement connectionist approach to robot path finding in non-maze-like environments. *Machine Learning*, 8:363–395, 1992.

[126] M. Minsky and S. Papert. *Perceptrons*. MIT Press, Cambridge, MA, 1988.

[127] T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[128] A. Moffat. *Compression and Coding Algorithms*. Kluwer Academic Publishers, 2002. ISBN 0-7923-7668-4.

[129] C. Monteleoni and T. Jaakkola. Online learning of non-stationary sequences. *Advances in Neural Information Processing Systems*, 16: 1093–1100, 2003.

[130] A. W. Moore. Reinforcement learning in factories: the auton project (abstract). In *Proc. 13th International Conference on Machine Learning*, page 556. Morgan Kaufmann, 1996.

[131] A. W. Moore and C. G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13: 103–130, 1993.

[132] D. E. Moriarty and R. Miikkulainen. Efficient reinforcement learning through symbiotic evolution. *Machine Learning*, 22:11–32, 1996.

[133] J. Morimoto and K. Doya. Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. In *Proc. 17th International Conf. on Machine Learning*, pages 623–630. Morgan Kaufmann, San Francisco, CA, 2000.

[134] R. Munos. A convergent reinforcement learning algorithm in the continuous case: the finite-element reinforcement learning. In *Proc. 13th International Conference on Machine Learning*, pages 337–345. Morgan Kaufmann, 1996.

[135] R. Munos. A study of reinforcement learning in the continuous case by the means of viscosity solutions. *Machine Learning*, 40(3): 265–299, 2000.

[136] B. K. Natarajan and P. Tadepalli. Two new frameworks for learning. In *Proc. of the 5th International Conference on Machine Learning*, pages 402–415, San Mateo, CA, June 1988. published by Morgan Kaufmann.

[137] A. Y. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *Proc. 17th International Conf. on Machine Learning*, pages 663–670. Morgan Kaufmann, San Francisco, CA, 2000.

[138] D. Ormoneit and Ś. Sen. Kernel-based reinforcement learning. *Machine Learning*, 49(2-3):161–178, 2002.

[139] M. D. Pendrith and M. J. McGarity. An analysis of direct reinforcement learning in non-Markovian domains. In *Proc. 15th International Conf. on Machine Learning*, pages 421–429. Morgan Kaufmann, San Francisco, CA, 1998.

[140] M. D. Pendrith and M. R. K. Ryan. Actual return reinforcement learning versus temporal differences: some theoretical and experimental results. In *Proc. 13th International Conference on Machine Learning*, pages 373–381. Morgan Kaufmann, 1996.

[141] T. J. Perkins and A. G. Barto. Lyapunov-constrained action sets for reinforcement learning. In *Proc. 18th International Conf. on Machine Learning*, pages 409–416. Morgan Kaufmann, San Francisco, CA, 2001.

[142] T. J. Perkins and A. G. Barto. Lyapunov design for safe reinforcement learning. *Journal of Machine Learning Research*, 3:803–832, 2002.

[143] J. Poland. FPL analysis for adaptive bandits. In *Stochastic Algorithms: Foundations and Applications, Third International Symposium, SAGA 2005, Moscow, Russia, October 2005, Proceedings*, volume 3777 of *Lecture Notes in Computer Science*, pages 58–69. Springer, 2005.

[144] D. Precup and R. S. Sutton. Exponentiated gradient methods for reinforcement learning. In *Proc. 14th International Conference on Machine Learning*, pages 272–277. Morgan Kaufmann, 1997.

[145] B. Price and C. Boutilier. Implicit imitation in multiagent reinforcement learning. In *Proc. 16th International Conf. on Machine Learning*, pages 325–334. Morgan Kaufmann, San Francisco, CA, 1999.

[146] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, issue 2, pages 257–285, 1989.

[147] J. Randløv. Shaping in reinforcement learning by changing the physics of the problem. In *Proc. 17th International Conf. on Machine Learning*, pages 767–774. Morgan Kaufmann, San Francisco, CA, 2000.

[148] J. Randløv and P. Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. In *Proc. 15th International Conf. on Machine Learning*, pages 463–471. Morgan Kaufmann, San Francisco, CA, 1998.

[149] J. Randløv, A. G. Barto, and M. T. Rosenstein. Combining reinforcement learning with a local control algorithm. In *Proc. 17th International Conf. on Machine Learning*, pages 775–782. Morgan Kaufmann, San Francisco, CA, 2000.

[150] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

[151] J. Rennie and A. K. McCallum. Using reinforcement learning to spider the web efficiently. In *Proc. 16th International Conf. on Machine Learning*, pages 335–343. Morgan Kaufmann, San Francisco, CA, 1999.

[152] S. I. Reynolds. Adaptive resolution model-free reinforcement learning: Decision boundary partitioning. In *Proc. 17th International Conf. on Machine Learning*, pages 783–790. Morgan Kaufmann, San Francisco, CA, 2000.

[153] C. Richter and J. Stachowiak. Knowledge propagation in model-based reinforcement learning tasks. In *Proc. 17th International Conf. on Machine Learning*, pages 791–798. Morgan Kaufmann, San Francisco, CA, 2000.

[154] J. Rissanen. *Stochastic Complexity in Statistical Inquiry*, volume 15 of *Series in Computer Science*. World Scientific, 1989.

[155] R. L. Rivest and Y. Yin. Simulation results for a new two-armed bandit heuristic. In S. J. Hanson, G. A. Drastal, and R. L. Rivest, editors, *Computational Learning Theory and Natural Learning Systems*, volume I: Constraints and Prospects, chapter 17, pages 477–486. MIT Press, 1994. Earlier version in 1990 Conference on Computation Learning and Natural Learning at Princeton.

[156] M. Ryan and M. Reid. Learning to fly: An application of hierarchical reinforcement learning. In *Proc. 17th International Conf. on Machine Learning*, pages 807–814. Morgan Kaufmann, San Francisco, CA, 2000.

[157] M. R. K. Ryan and M. D. Pendrith. RL-TOPs: an architecture for modularity and re-use in reinforcement learning. In *Proc. 15th International Conf. on Machine Learning*, pages 481–487. Morgan Kaufmann, San Francisco, CA, 1998.

[158] M. Salganicoff and L. H. Ungar. Active exploration and learning in real-valued spaces using multi-armed bandit allocation indices. In *Proc. 12th International Conference on Machine Learning*, pages 480–487. Morgan Kaufmann, 1995.

[159] M. Sato and S. Kobayashi. Average-reward reinforcement learning for variance penalized Markov decision problems. In *Proc. 18th International Conf. on Machine Learning*, pages 473–480. Morgan Kaufmann, San Francisco, CA, 2001.

[160] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[161] A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer-Verlag, Berlin, 2003.

[162] G. Shafer, A. Shen, N. Vereshchagin, and V. Vovk. Test martingales, Bayes factors, and p-values. *Statistical Science*, 2011. To appear. Preprint available as arXiv:0912.4269.

[163] J. Shawe-Taylor, P. L. Bartlett, R. C. Williamson, and M. Anthony. A framework for structural risk minimization. In *Proc. 9th Annu. Conf. on Comput. Learning Theory*, pages 68–76. ACM Press, New York, NY, 1996.

[164] S. P. Singh and R. S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22:123–158, 1996.

[165] W. D. Smart and L. P. Kaelbling. Practical reinforcement learning in continuous spaces. In *Proc. 17th International Conf. on Machine Learning*, pages 903–910. Morgan Kaufmann, San Francisco, CA, 2000.

[166] P. Stone and R. S. Sutton. Scaling reinforcement learning toward RoboCup soccer. In *Proc. 18th International Conf. on Machine Learning*, pages 537–544. Morgan Kaufmann, San Francisco, CA, 2001.

[167] M. Strens. A Bayesian framework for reinforcement learning. In *Proc. 17th International Conf. on Machine Learning*, pages 943–950. Morgan Kaufmann, San Francisco, CA, 2000.

[168] R. S. Sutton. Reinforcement learning architectures for animats. In *First International Conference on Simulation of Adaptive Behavior*, 1991.

[169] R. S. Sutton. Open theoretical questions in reinforcement learning. In *Computational Learning Theory, 4th European Conference, EuroCOLT '99, Nordkirchen, Germany, March 29-31, 1999, Proceedings*, volume 1572 of *Lecture Notes in Artificial Intelligence*, pages 11–17. Springer, 1999.

[170] P. Tadepalli and T. G. Dietterich. Hierarchical explanation-based reinforcement learning. In *Proc. 14th International Conference on Machine Learning*, pages 358–366. Morgan Kaufmann, 1997.

[171] P. Tadepalli and D. Ok. Scaling up average reward reinforcement learning by approximating the domain models and the value

function. In *Proc. 13th International Conference on Machine Learning*, pages 471–479. Morgan Kaufmann, 1996.

[172] E. Takimoto and M. Warmuth. The last-step minimax algorithm. In *Proceedings of the 13th Annual Conference on Computational Learning Theory*, pages 100–106, 2000.

[173] E. Takimoto and M. K. Warmuth. Path kernels and multiplicative updates. *Journal of Machine Learning Research*, 4:773–818, 2003. ISSN 1532-4435.

[174] A. Teller and M. Veloso. Efficient learning trough evolution: Neural programming and internal reinforcement. In *Proc. 17th International Conf. on Machine Learning*, pages 959–966. Morgan Kaufmann, San Francisco, CA, 2000.

[175] H. Tong and T. X. Brown. Reinforcement learning for call admission control and routing under quality of service constraints in multimedia networks. *Machine Learning*, 49(2-3):111–139, 2002.

[176] J. N. Tsitsiklis. A lemma on the multiarmed bandit problem. *IEEE Transactions on Automatic Control*, AC-31(6), June 1986.

[177] T. van Erven, P. Grünwald, and S. de Rooij. Catching up faster by switching sooner: a prequential solution to the AIC-BIC dilemma. Submitted. Preprint available as arXiv:0807.1005., 2008.

[178] T. van Erven, P. D. Grünwald, and S. de Rooij. Catching up faster in Bayesian model selection and model averaging. In *Advances in Neural Information Processing Systems 20 (NIPS 2007)*, 2008.

[179] V. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.

[180] P. A. Volf and F. M. Willems. Switching between two universal source coding algorithms. In *Proceedings of the Data Compression Conference, Snowbird, Utah*, pages 491–500, 1998.

[181] V. Vovk. Aggregating strategies. In *Proceedings of the third Annual Conference on Computational Learning Theory (COLT)*, pages 371–383, 1990.

[182] V. Vovk. A game of prediction with expert advice. *Journal of Computer and System Sciences*, 56:153–173, 1998.

[183] V. Vovk. Derandomizing stochastic prediction strategies. *Machine Learning*, 35:247–282, 1999.

[184] D. J. Ward and D. J. C. MacKay. Artificial intelligence: Fast hands-free writing by gaze direction. *Nature*, 418(6900):838–841, Aug. 2002.

[185] M. K. Warmuth and D. Kuzmin. Randomized online PCA algorithms with regret bounds that are logarithmic in the dimension. *Journal of Machine Learning Research*, 9:2287–2320, Oct. 2008.

[186] M. K. Warmuth, K. Glocer, and S. Vishwanathan. Entropy regularized LPBoost. In Y. Freund, L. Györfi, G. Turán, and T. Zeugmann, editors, *Proceedings of the 19th International Conference on Algorithmic Learning Theory (ALT '08)*, pages 256–271. Springer-Verlag, Oct. 2008.

[187] M. Wiering. Multi-agent reinforcement learning for traffic light control. In *Proc. 17th International Conf. on Machine Learning*, pages 1151–1158. Morgan Kaufmann, San Francisco, CA, 2000.

[188] M. A. Wiering. Reinforcement learning in dynamic environments using instantiated information. In *Proc. 18th International Conf. on Machine Learning*, pages 585–592. Morgan Kaufmann, San Francisco, CA, 2001.

[189] F. Willems, Y. Shtarkov, and T. Tjalkens. The context tree weighting method: basic properties. *IEEE Transactions on Information Theory*, 41(3):653–664, 1995.

[190] F. M. Willems. Coding for a binary independent piecewise-identically distributed source. *IEEE Transactions on Information Theory*, 42(6):2210–2217, Nov. 1996.

[191] J. L. Wyatt. Exploration control in reinforcement learning using optimistic model selection. In *Proc. 18th International Conf. on Machine Learning*, pages 593–600. Morgan Kaufmann, San Francisco, CA, 2001.

Index

Samenvatting

In dit proefschrift bestuderen we *machine learning*: het automatisch vinden en benutten van regelmatigheden in data.

We kunnen regelmatigheden die we hebben geïdentificeerd in *objecten* gebruiken om het verleden te verklaren (b.v. archeologie, rechtspraak), en regelmatigheden die we hebben gevonden in *processen* om de toekomst te voorspellen (b.v. het weer, beurskoersen) en om ons handelen te leiden.

Dankzij alom beschikbare rekencapaciteit zijn machine learning algoritmen tegenwoordig overal doorgedrongen. Zij beheren bijvoorbeeld financiële portfolios, managen het energiebesparingsbeleid van draagbare apparatuur, bevelen films alsook advertenties aan gebaseerd op persoonlijke voorkeuren, en zij vormen het hart van de best beschikbare datacompressieprogrammatuur.

Dit proefschrift is een bijdrage aan de theorie van *online learning*, een tak van machine learning die sequentiële beslissingsproblemen met onmiddellijke terugkoppeling bestudeert.

In het bijzonder bestuderen we de opzet genaamd *voorspellen met expertadvies*. Het is hier onze taak om een reeks data te voorspellen. Elke ronde raadplegen we hiertoe eerst een set experts. Daarna combineren we hun adviezen en leveren zo onze eigen voorspelling van de volgende uitkomst. Tenslotte wordt de volgende uitkomst onthuld, en boeten we verlies in voor de discrepantie tussen onze voorspelling en de gerealiseerde uitkomst.

Het doel is om efficiënte algoritmen te bouwen met weinig *spijt*, d.w.z. het verschil tussen het ingeboete cumulatieve verlies van het algoritme en het verlies van de beste strategie, achteraf gekozen uit een vaste referentieklasse. In deze zin kunnen de strategieën in de referen-

tieklasse beschouwd worden als mogelijke patronen, en betekent het oplopen van weinig spijt dat geleerd wordt welke referentiestrategie de data het beste modelleert. Het belangrijkste verschil tussen de leerproblemen die we beschouwen is de complexiteit van de referentieset.

Algoritmen voor het voorspellen met expertadvies hebben reeds legio toepassingen, waaronder classificatie, regressie, hypothesetoetsen, modelselectie, datacompressie, gokken en investeren in de aandelenbeurs.

In hoofdstuk 2 geven we een speltheoretische analyse van het simpelste online learning probleem, het voorspellen van een reeks binaire uitkomsten onder de 0/1 verliesmaat met behulp van twee experts. Voor dit simpele probleem berekenen we de minimax, d.w.z. speltheoretisch optimale spijt, en laten zien hoe de optimale strategie efficiënt te implementeren is. Daarna geven we speciale aandacht aan het geval dat een expert erg goed is. We sluiten af met een nieuw resultaat: het optimale algoritme voor wedijveren met de set meta-experts die wisselen tussen de twee basisexperts.

In hoofdstuk 3 laten we zien hoe modellen voor voorspellen met expertadvies beknopt en helder kunnen worden gedefinieerd met gebruik van hidden Markov modellen (HMMs); standaardalgoritmen kunnen dan worden gebruikt om efficiënt uit te rekenen hoe de voorspellingen van de experts gewogen moeten worden. We concentreren ons op algoritmen voor het volgen van de beste expert. Voor deze taak volgen de strategieën in de referentieset steeds het advies van een enkele expert, maar welke expert dit is kan in verloop van tijd veranderen. We herbeschrijven bestaande modellen als HMMs, beginnend bij het fixed share algoritme, leiden de uitvoeringstijd en spijtbovengrens overnieuw af, en bespreken de onderlinge verbanden. We beschrijven ook drie nieuwe modellen voor het wisselen tussen experts.

In hoofdstuk 4 breiden we de opzet uit naar het volgen van de beste *lerende* expert. Gebruikelijke experts geven elke ronde een advies over de volgende uitkomst. Lerende experts kunnen daarintegen bevraagd worden gegeven elke mogelijke subset van de data uit het verleden. Deze extra mogelijkheid staat ter beschikking van zowel het algoritme als van de referentiestrategieën. Het behalen van weinig spijt betekent nu te leren de rondes te partitioneren, en de beste lerende expert te trainen en te volgen binnen elke cel van de partitie. We geven efficiënte algoritmen met weinig spijt voor het volgen van lerende experts die zelf

uitgedrukt kunnen worden d.m.v. de expert HMMs uit hoofdstuk 3.

In hoofdstuk 5 beschouwen we referentiestrategieën die wisselen tussen twee experts gebaseerd op hun *cumulatieve verlies* in plaats van op de *tijd*. Dit hoofdstuk is geformuleerd in financiele termen om de presentatie intuïtiever te maken. We presenteren een simpel online handelsalgoritme dat fluctuaties uitbuit in de eenheidsprijs van een activum. In plaats van de opbrengst te analyseren in het ongunstigste geval onder zekere aannamen, bewijzen wij een nieuwe, aannamevrije opbrengstgarantie die is geparametriseerd ofwel met de echte dynamiek van de prijs van het activum, danwel met een versimpeling daarvan.

We bespreken toepassingen van de resultaten op voorspellen met expertadvies, datacompressie en hypothesetoetsen.

In hoofdstuk 6 beschouwen we voorspellen met *gestructureerde concepten*. Elke ronde keizen we een concept dat is opgebouwd uit componenten. Het verlies van een concept is de som van de verliezen van diens componenten. Terwijl de verliezen van verschillende *componenten* onafhankelijk zijn, zijn de verliezen van verschillende *concepten* juist hoogst gerelateerd. We ontwikkelen een online algoritme, *Component Hedge* genaamd, dat deze afhankelijkheden uitbuit, en daardoor de zogenaamde *bereikfactor* vermijdt, die optreedt als de afhankelijkheden worden genegeerd. We laten zien dat Component Hedge optimale spijtgaranties heeft voor een grote verscheidenheid aan gestructureerde conceptklassen.

Abstract

In this dissertation we study *machine learning*: the automated discovery and exploitation of regularities in data. We may use regularities identified in *objects* to explain the past (e.g. archaeology, justice), as well as regularities found in *processes* to predict the future (e.g. weather, stock market) and guide our actions.

With ubiquitous computational resources, machine learning algorithms have become pervasive. For example, they manage financial portfolios and power-saving policy, provide personalised movie recommendations as well as advertisements, and form the core of state-of-the-art data compression software.

This dissertation develops the theory of *online learning*, a branch of machine learning that investigates sequential decision problems with immediate feedback. In particular, we study the setting called *prediction with expert advice*. Our task is to predict a sequence of data. Each trial, we may first consult a given set of experts. We then combine their advice and issue our prediction of the next outcome. Finally, the next outcome is revealed, and we incur loss based on the discrepancy between our prediction and it.

The goal is to build efficient algorithms with small *regret*, i.e. the difference between the incurred cumulative loss and the loss of the best strategy in hindsight from a fixed reference class. In this sense, the strategies in the reference class are the patterns, and achieving small regret means *learning* which reference strategy best models the data. The main difference between the learning problems we consider is the complexity of the reference set. Algorithms for prediction with expert advice have many applications including classification, regression, hypothesis testing, model selection, data compression, gambling and

investing in the stock market.

In Chapter 2 we give a game-theoretic analysis of the simplest online learning problem, the prediction of a sequence of binary outcomes under 0/1 loss with the help of two experts. For this simple problem, we compute the minimax, i.e. game-theoretically optimal, regret, and show how to implement the optimal strategy efficiently. We then give special attention to the case that one of the experts is good. We conclude with a new result: the optimal algorithm for competing with the set of meta-experts that switch between the two basic experts.

In Chapter 3 we show how models for prediction with expert advice can be defined concisely and clearly using hidden Markov models (HMMs); standard algorithms can then be used to efficiently calculate how the expert predictions should be weighted. We focus on algorithms for *tracking the best expert*. Here the strategies in the reference set follow the advice of a single expert, but this expert may change between trials. We cast existing models as HMMs, starting from the fixed share algorithm, recover the running times and regret bounds for each algorithm, and discuss how they are related. We also describe three new models for switching between experts.

In Chapter 4 we extend the setting to tracking the best *learning* expert. Whereas vanilla experts can be tapped for advice about the current trial, learning experts may be queried for advice *given each possible subset* of the past data. This additional power is available to both the algorithm and the reference strategies. Achieving small regret thus means learning how to partition the trials, and which learning expert to train and follow within each partition cell. We give efficient algorithms with small regret for tracking learning experts that can themselves be formalised using the expert HMMs of Chapter 3.

In Chapter 5 we consider reference strategies that switch between two experts based on their *cumulative loss* instead of on *time*. This chapter is formulated in financial terms to make the presentation more intuitive. We present a simple online two-way trading algorithm that exploits fluctuations in the unit price of an asset. Rather than analysing worst-case performance under some assumptions, we prove a novel, unconditional performance bound that is parameterised either by the actual dynamics of the price of the asset, or by a simplifying model thereof. We discuss application of the results to prediction with expert advice, data compression and hypothesis testing.

In Chapter 6 we consider prediction with *structured concepts*. Each round we select a concept, which is composed of components. The loss of a concept is the sum of the losses of its components. Whereas the losses of different components are independent, the losses of different *concepts* are highly related. We develop an online algorithm, called *Component Hedge* that exploits this dependence, and thereby avoids the so called *range factor* that arises when the dependences are ignored. We show that Component Hedge has optimal regret bounds for a large variety of structured concept classes.

Curriculum Vitae

Personalia

Surname	Koolen
Name	Wouter Michiel
Birth	2 July 1982, Groningen (The Netherlands)
Nationality	Dutch, Swiss
Address	Valckenierstraat 45 A
	1018 XE Amsterdam
	The Netherlands
Phone	+31 6 49682755
e-mail	wmkoolen@cwi.nl
Homepage	http://www.cwi.nl/~wmkoolen

Fellowships & Education

Postdoc Computer Learning Research Centre
February 2011 - January 2013 Royal Holloway, London, UK
Project: *Game-Theoretically Optimal Online Learning: From Conflicting Advice to High-Quality Decisions*

Ph.D. Centrum Wiskunde en Informatica
January 2007 - January 2011 Amsterdam, The Netherlands
Promotores: prof. Paul M.B. Vitányi, prof. Peter D. Grünwald
Dissertation: *Combining Strategies Efficiently: High-Quality Decisions from Conflicting Advice*

Master of Science in Logic Universiteit van Amsterdam
September 2004 - December 2006 Amsterdam, The Netherlands
Supervisor: prof. Peter D. Grünwald
Thesis: *Discovering the Truth by Conducting Experiments*
Cum Laude

Honors & Awards

My propaedeutic (UvA, 2001) and bachelor's (UvA, 2003) degrees in both Computer Science and Artificial Intelligence, my Maîtrise (Université de Nice-Sophia Antipolis, France, 2004) in Computer Science and my Master's degree in Logic (UvA, 2006) were all obtained cum laude.

In 2010, the Netherlands Organisation for Scientific Research (NWO) honored my project proposal *Game-Theoretically Optimal Online Learning: From Conflicting Advice to High-Quality Decisions* with a RUBICON grant, funding two years of post-doctoral research with prof. Vladimir Vovk at the Computer Learning Research Centre at Royal Holloway, University of London.

Publications

Journal

- H. Buhrman, P. T. S. van der Gulik, S. M. Kelk, W. M. Koolen, and L. Stougie. Some mathematical refinements concerning error minimization in the genetic code. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 2010. Accepted.
- E. G. Daylight, W. M. Koolen, and P. M. Vitányi. Time-bounded incompressibility of compressible strings and sequences. *Information Processing Letters (IPL)*, 109(18):1055 – 1059, Aug. 2009.

Conference

- W. M. Koolen and S. de Rooij. Switching investments. In M. Hutter, F. Stephan, V. Vovk, and T. Zeugman, editors, *Proceedings of*

the 21st International Conference on Algorithmic Learning Theory (ALT 2010), LNAI 6331, pages 239–254. Springer, Heidelberg, Oct. 2010.

- W. M. Koolen, M. K. Warmuth, and J. Kivinen. Hedging structured concepts. In *Proceedings of the 23rd Annual Conference on Learning Theory (COLT 2010)*, pages 93–105, June 2010.
- M. Ziegler and W. M. Koolen. Kolmogorov complexity theory over the reals. *Electronic Notes in Theoretical Computer Science (ENTCS)*, 221:153–169, Dec. 2008.
- W. M. Koolen and S. de Rooij. Combining expert advice efficiently. In R. Servedio and T. Zang, editors, *Proceedings of the 21st Annual Conference on Learning Theory (COLT 2008)*, pages 275–286, June 2008.

Technical Report

- W. M. Koolen and T. van Erven. Switching between hidden Markov models using Fixed Share. *Computing Research Repository (CoRR)*, abs/1008.4532, Feb. 2010.
- H. Buhrman, P. T. S. van der Gulik, S. M. Kelk, W. M. Koolen, and L. Stougie. Some mathematical refinements concerning error minimization in the genetic code. *arXiv*, abs/0909.1442, Sept. 2009.
- W. M. Koolen and T. van Erven. Freezing and sleeping: Tracking experts that learn by evolving past posteriors. *Computing Research Repository (CoRR)*, abs/1008.4654, Feb. 2009.
- E. G. Daylight, W. M. Koolen, and P. M. B. Vitányi. On time-bounded incompressibility of compressible strings and sequences. *Computing Research Repository (CoRR)*, abs/0809.2965, Sept. 2008.
- M. Ziegler and W. M. Koolen. Kolmogorov complexity theory over the reals. *Computing Research Repository (CoRR)*, abs/0802.2027, Feb. 2008.
- W. M. Koolen and S. de Rooij. Combining expert advice efficiently. *Computing Research Repository (CoRR)*, abs/0802.2015, Feb. 2008.
- W. M. Koolen. Temporary unavailability logic and general modification logic. *ILLC Prepublication Series*, Jan. 2008.

Extended Abstracts (Local Dissemination)

- W. M. Koolen and T. van Erven. Freezing and sleeping: Tracking experts that learn by evolving past posteriors. In *Proceedings of the 18th Annual Belgian-Dutch Conference on Machine Learning (Bene-Learn 2009)*, pages 91–92, May 2009.
- W. M. Koolen and S. de Rooij. Combining expert advice efficiently. In A. Nijholt, M. Pantic, M. Poel, and H. Hondorp, editors, *Proceedings of the twentieth Belgian-Dutch Conference on Artificial Intelligence (BNAIC 2008)*, pages 323–324, Oct. 2008.

Master's Thesis

- W. M. Koolen. Discovering the truth by conducting experiments. Master's thesis, Institute of Logic, Language and Computation, Universiteit van Amsterdam, Dec. 2006.

Teaching

Kolmogorov Complexity FNWI, Universiteit van Amsterdam
Spring 2007, Spring 2008
This graduate level course was taught yearly by prof. Paul Vitányi at the Universiteit van Amsterdam. I assisted Paul by teaching the homework/lab session and grading the homework and exams, and tutored Edgar G. Daylight during the term project that he undertook, which resulting in the second journal publication above.

Teaching Assistant FNWI, Universiteit van Amsterdam
September 2001 - December 2002
I assisted the following undergraduate level courses:
• Logisch Programmeren (Introduction to Prolog for A.I. students),
• Kennis en Interactie and
• Informatie en Informatieverwerking.

Employment History

Software developer AMSTEL Instituut, Universiteit van Amsterdam
January 2002 - March 2007

I designed and implemented the SIM-PL software package, an educational tool for the design and simulation of digital components and digital circuits. SIM-PL simulates the whole spectrum from simple gates to super-scalar processors, and gives a timing-accurate insight into the internal workings of these circuits.

SIM-PL is currently used in the courses Digitale Technieken and Computer Architectuur at the Universiteit van Amsterdam.
`http://www.science.uva.nl/amstel/SIM-PL/`

Other interests

- Hiking, trekking, climbing, survival. Former Scouting member.
- Fantasy role-playing, Dungeons and Dragons.
- Bach, Mozart and symphonic metal.
- Medieval castles.
- Cooking.
- Programming.

Titles in the ILLC Dissertation Series:

ILLC DS-2006-01: **Troy Lee**
Kolmogorov complexity and formula size lower bounds

ILLC DS-2006-02: **Nick Bezhanishvili**
Lattices of intermediate and cylindric modal logics

ILLC DS-2006-03: **Clemens Kupke**
Finitary coalgebraic logics

ILLC DS-2006-04: **Robert Špalek**
Quantum Algorithms, Lower Bounds, and Time-Space Tradeoffs

ILLC DS-2006-05: **Aline Honingh**
The Origin and Well-Formedness of Tonal Pitch Structures

ILLC DS-2006-06: **Merlijn Sevenster**
Branches of imperfect information: logic, games, and computation

ILLC DS-2006-07: **Marie Nilsenova**
Rises and Falls. Studies in the Semantics and Pragmatics of Intonation

ILLC DS-2006-08: **Darko Sarenac**
Products of Topological Modal Logics

ILLC DS-2007-01: **Rudi Cilibrasi**
Statistical Inference Through Data Compression

ILLC DS-2007-02: **Neta Spiro**
What contributes to the perception of musical phrases in western classical music?

ILLC DS-2007-03: **Darrin Hindsill**
It's a Process and an Event: Perspectives in Event Semantics

ILLC DS-2007-04: **Katrin Schulz**
Minimal Models in Semantics and Pragmatics: Free Choice, Exhaustivity, and Conditionals

ILLC DS-2007-05: **Yoav Seginer**
Learning Syntactic Structure

ILLC DS-2008-01: **Stephanie Wehner**
Cryptography in a Quantum World

ILLC DS-2008-02: **Fenrong Liu**
Changing for the Better: Preference Dynamics and Agent Diversity

ILLC DS-2008-03: **Olivier Roy**
Thinking before Acting: Intentions, Logic, Rational Choice

ILLC DS-2008-04: **Patrick Girard**
Modal Logic for Belief and Preference Change

ILLC DS-2008-05: **Erik Rietveld**
Unreflective Action: A Philosophical Contribution to Integrative Neuroscience

ILLC DS-2008-06: **Falk Unger**
Noise in Quantum and Classical Computation and Non-locality

ILLC DS-2008-07: **Steven de Rooij**
Minimum Description Length Model Selection: Problems and Extensions

ILLC DS-2008-08: **Fabrice Nauze**
Modality in Typological Perspective

ILLC DS-2008-09: **Floris Roelofsen**
Anaphora Resolved

ILLC DS-2008-10: **Marian Counihan**
Looking for logic in all the wrong places: an investigation of language, literacy and logic in reasoning

ILLC DS-2009-01: **Jakub Szymanik**
Quantifiers in TIME and SPACE. Computational Complexity of Generalized Quantifiers in Natural Language

ILLC DS-2009-02: **Hartmut Fitz**
Neural Syntax

ILLC DS-2009-03: **Brian Thomas Semmes**
A Game for the Borel Functions

ILLC DS-2009-04: **Sara L. Uckelman**
Modalities in Medieval Logic

ILLC DS-2009-05: **Andreas Witzel**
Knowledge and Games: Theory and Implementation

ILLC DS-2009-06: **Chantal Bax**
Subjectivity after Wittgenstein. Wittgenstein's embodied and embedded subject and the debate about the death of man.

ILLC DS-2009-07: **Kata Balogh**
Theme with Variations. A Context-based Analysis of Focus

ILLC DS-2009-08: **Tomohiro Hoshi**
Epistemic Dynamics and Protocol Information

ILLC DS-2009-09: **Olivia Ladinig**
Temporal expectations and their violations

ILLC DS-2009-10: **Tikitu de Jager**
"Now that you mention it, I wonder…": Awareness, Attention, Assumption

ILLC DS-2009-11: **Michael Franke**
Signal to Act: Game Theory in Pragmatics

ILLC DS-2009-12: **Joel Uckelman**
More Than the Sum of Its Parts: Compact Preference Representation Over Combinatorial Domains

ILLC DS-2009-13: **Stefan Bold**
Cardinals as Ultrapowers. A Canonical Measure Analysis under the Axiom of Determinacy.

ILLC DS-2010-01: **Reut Tsarfaty**
Relational-Realizational Parsing

ILLC DS-2010-02: **Jonathan Zvesper**
Playing with Information

ILLC DS-2010-03: **Cédric Dégremont**
The Temporal Mind. Observations on the logic of belief change in interactive systems

ILLC DS-2010-04: **Daisuke Ikegami**
Games in Set Theory and Logic

ILLC DS-2010-05: **Jarmo Kontinen**
Coherence and Complexity in Fragments of Dependence Logic

ILLC DS-2010-06: **Yanjing Wang**
Epistemic Modelling and Protocol Dynamics

ILLC DS-2010-07: **Marc Staudacher**
Use theories of meaning between conventions and social norms

ILLC DS-2010-08: **Amélie Gheerbrant**
Fixed-Point Logics on Trees

ILLC DS-2010-09: **Gaëlle Fontaine**
Modal Fixpoint Logic: Some Model Theoretic Questions

ILLC DS-2010-10: **Jacob Vosmaer**
Logic, Algebra and Topology. Investigations into canonical extensions, duality theory and point-free topology.

ILLC DS-2010-11: **Nina Gierasimczuk**
Knowing One's Limits. Logical Analysis of Inductive Inference

ILLC DS-2011-01: **Wouter M. Koolen**
Combining Strategies Efficiently: High-Quality Decisions from Conflicting Advice