



UvA-DARE (Digital Academic Repository)

Combining strategies efficiently: high-quality decisions from conflicting advice

Koolen, W.M.

Publication date
2011

[Link to publication](#)

Citation for published version (APA):

Koolen, W. M. (2011). *Combining strategies efficiently: high-quality decisions from conflicting advice*. [Thesis, fully internal, Universiteit van Amsterdam].

General rights

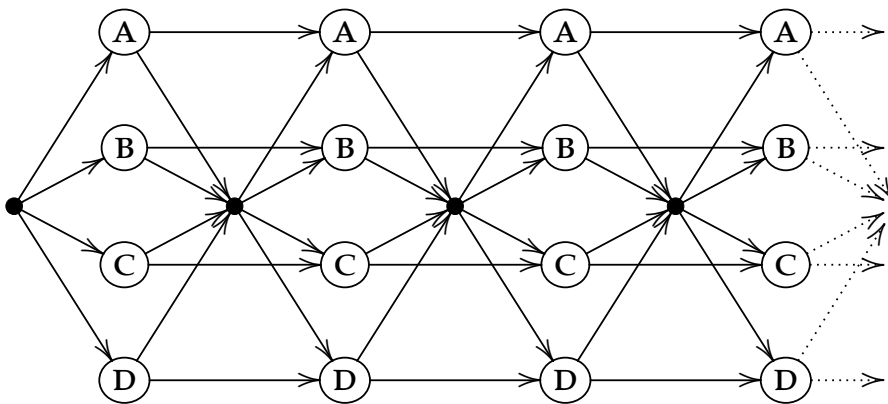
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations

If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: <https://uba.uva.nl/en/contact>, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

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 $\zeta \in \Xi$ issues a distribution $P_\zeta(\mathbf{x}_{t+1}|\mathbf{x}^t)$ on the next outcome \mathbf{x}_{t+1} given the previous observations $\mathbf{x}^t := x_1, \dots, x_t$. 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 $\mathbf{x}_{t+1} = x_{t+1}$ is observed, the expert suffers *logarithmic loss* $-\ln P_\zeta(x_{t+1}|\mathbf{x}^t)$. 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 ζ , our accumulated loss is

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

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 $P(\mathbf{x}^t, \zeta) = w(\zeta)P_\zeta(\mathbf{x}^t)$. 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 $P(x_{t+1}|x^t) = P(x^t, x_{t+1})/P(x^t)$, which defines a prediction strategy that combines those of the individual experts, or (c) the *posterior distribution* on the experts $P(\xi|x^t) = P_{\xi}(x^t)w(\xi)/P(x^t)$, 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 $O(kt)$ 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 ξ_1, ξ_2, \dots 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 $P(\xi_{t+1}|x^t)$.

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 $q^t = \langle q_1, \dots, q_t \rangle$. We also let q^ω range over Q^ω — the set of infinite sequences over Q — and write $q^\omega = \langle q_1, \dots \rangle$. 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 $\text{prob}(\mathcal{X})$ denote the set of all probability mass functions on \mathcal{X} . A (*prequential*) \mathcal{X} -forecasting system (PFS) is a function $P : \mathcal{X}^* \rightarrow \text{prob}(\mathcal{X})$ 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}^ω is defined by a function $P : \mathcal{X}^* \rightarrow [0, 1]$ 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 $\sum_{x \in \mathcal{X}} P(x_1, \dots, x_t, x) = P(x^t)$. 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 $\zeta \in \Xi$ predicts using a forecasting system P_ζ .

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

$$P_{\zeta^t}(x^t) := \prod_{i=1}^t P_{\zeta_i}(x_i|x^{i-1}).$$

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:

$$P_\pi(\zeta^t, x^t) := \pi(\zeta^t) P_{\zeta^t}(x^t). \quad (3.1)$$

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

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

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 $\theta \in \Theta$, 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 ζ^ω . Namely, at each moment in time the predictions of the experts are weighted according to the posterior probability of ζ_{t+1} , which can be computed as follows:

$$P_\pi(\zeta_{t+1}|x^t) = \frac{\sum_{\zeta^t} P_\pi(\zeta^t, x^t, \zeta_{t+1})}{\sum_{\zeta^t} P_\pi(\zeta^t, x^t)} = \frac{\sum_{\zeta^t} \pi(\zeta^t, \zeta_{t+1}) P_{\zeta^t}(x^t)}{\sum_{\zeta^t} \pi(\zeta^t) P_{\zeta^t}(x^t)}. \quad (3.3)$$

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

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

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 $\xi \in \Xi$:

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

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

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:

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

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

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

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

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 $\pi_{\text{mix},w}(\xi_{t+1} | \xi^t) = w(\xi_{t+1})$ 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

¹These 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 $\mathcal{O} = \{n, s\}$.

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

$$\mathfrak{H} = \left\langle Q, Q_p, P_o, P_\rightarrow, \langle P_{\downarrow q} \rangle_{q \in Q_p} \right\rangle$$

a *hidden Markov model* on \mathcal{O} if Q is a countable set, $Q_p \subseteq Q$, $P_o \in \text{prob}(Q)$, $P_\rightarrow : Q \rightarrow \text{prob}(Q)$ and $P_{\downarrow q}$ is an \mathcal{O} -forecasting system for each $q \in Q_p$.

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

We let S_q denote the support of $P_{\circ}(q)$, and call $q' \in S_q$ a *direct successor* of q . We abbreviate $P_{\circ}(q)(q')$ to $P(q \rightarrow q')$. 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_p^λ its subsequence of productive states. We denote the elements of q_p^λ by q_1^p, q_2^p etc. We call an HMM *continuous* if q_p^ω 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 $\lambda \geq 1$

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

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

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

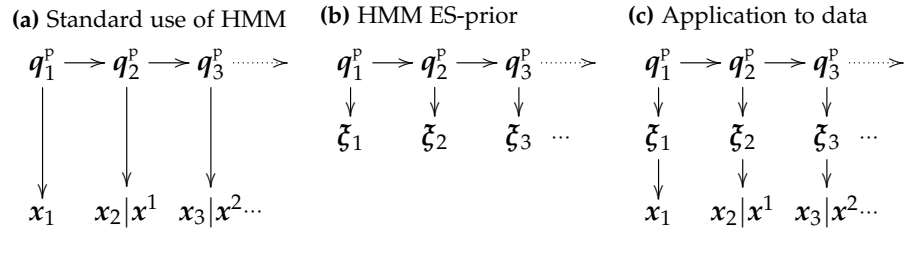
The value of $P_{\mathfrak{H}}$ at arguments o^t, q^λ that do not fulfil the condition above is determined by the additivity axiom of probability.

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

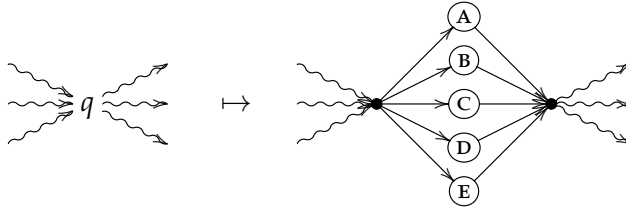
Figure 3.1 HMMs. q_i^p , ζ_i and x_i are the i th productive state, expert and observation.



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_i^p on the sequence of previously produced experts ζ^{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 $q \in Q_p$ by the following gadget:

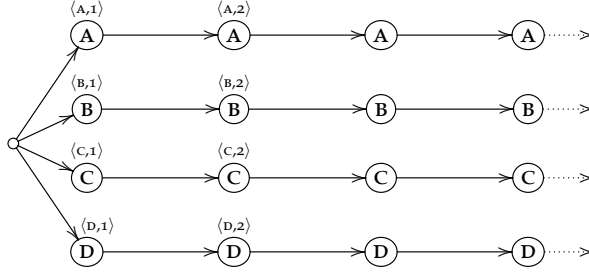


In the left diagram, the state q has distribution $P_{\downarrow q}$ on outcomes $\mathcal{O} = \{A, \dots, E\}$. In the right diagram, the leftmost silent state has transition probability $P_{\downarrow q}(o)$ to a state that deterministically outputs outcome o . We call a HMM $\langle Q, Q_p, P_o, P_{\rightarrow}, \Lambda \rangle$ on \mathcal{O} *deterministic* if Λ maps productive states to fixed outcomes, i.e. it is a function $\Lambda : Q_p \rightarrow \mathcal{O}$. 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. \textcircled{A} . We draw an arrow from N_q to $N_{q'}$ if q' is a direct successor of q . We often reify the initial distribution P_o 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 $P(q \rightarrow q')$ is not displayed in the graph.

We are now ready to give the deterministic HMMs that correspond

Figure 3.2 Standard Bayesian mixture $\text{BAYES}[\Xi, w]$ 

$$\text{BAYES}[\Xi, w] = \langle Q, Q_p, P_o, P_\rightarrow, \Lambda \rangle$$

$$Q = Q_p = \Xi \times \mathbb{Z}_+$$

$$\Lambda(\zeta, t) = \zeta \quad P_o(\zeta, 1) = w(\zeta)$$

$$P(\langle \zeta, t \rangle \rightarrow \langle \zeta, t+1 \rangle) = 1$$

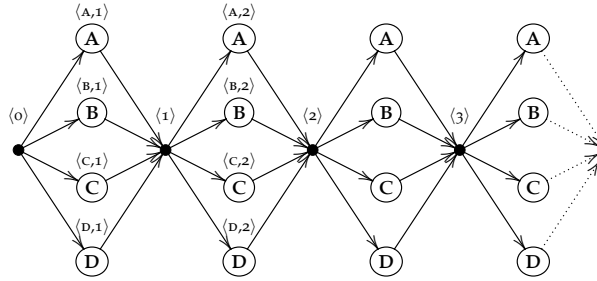
to the ES-priors of our running examples: Bayesian mixtures and elementwise mixtures with fixed parameters.

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 $\text{BAYES}[\Xi, w]$ as defined in Figure 3.2. Since the HMM computes the Bayesian mixture, (3.4) tells us that the regret (log-loss overhead) of $\text{BAYES}[\Xi, w]$ w.r.t. each expert $\zeta \in \Xi$ is bounded for all data x^t by

$$\ln \frac{P_\zeta(x^t)}{P_{\text{BAYES}[\Xi, w]}(x^t)} \leq -\ln w(\zeta). \quad (3.6)$$

In particular this bound holds for $\hat{\zeta} = \arg\max_{\zeta} P_\zeta(x^t)$, 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 $O(kt)$ using its definition (3.4). We show in Section 3.3.4 that computing $P_{\text{BAYES}[\Xi, w]}(x^t)$ has exactly the same running time. \diamond

3.3.4. EXAMPLE (HMM for Elementwise Mixtures). The deterministic HMM $\text{EM}[\Xi, w]$ 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 P_\rightarrow is shorthand

Figure 3.3 Fixed elementwise mixture $\text{EM}[\Xi, w]$ 

$$\begin{aligned} \text{EM}[\Xi, w] &= \langle Q, Q_p, P_o, P_\rightarrow, \Lambda \rangle \\ Q &= Q_s \cup Q_p \quad Q_s = \mathbb{N} \quad Q_p = \Xi \times \mathbb{Z}_+ \\ P_o(0) &= 1 \quad \Lambda(\xi, t) = \xi \\ P \left(\begin{array}{l} \langle t \rangle \rightarrow \langle \xi, t+1 \rangle \\ \langle \xi, t \rangle \rightarrow \langle t \rangle \end{array} \right) &= \begin{pmatrix} w(\xi) \\ 1 \end{pmatrix} \end{aligned}$$

for one 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 $P_{\text{EM}[\Xi, w]}(x^t)$ to be calculated in $O(kt)$ time as well. \diamond

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 $\mathfrak{H} = \langle Q, Q_p, P_o, P_\rightarrow, \Lambda \rangle$ 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 $\mathbb{X} := \langle Q, Q_p, P_o, P_{\rightarrow}, \langle P_{\Lambda(q)} \rangle_{q \in Q_p} \rangle$ 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 $P(x_{t+1}|x^t)$ 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 P_{\rightarrow} , one may consider a collection of transition functions $\langle P_{\rightarrow, \theta} \mid \theta \in \Theta \rangle$ 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, \dots . 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 P_\circ is on productive states. The *unfolding* of \mathfrak{H} is the HMM

$$\mathcal{U}(\mathfrak{H}) := \langle Q^u, Q_p^u, P_\circ^u, P_\rightarrow^u, \langle P_{\downarrow q}^u \rangle_{q \in Q^u} \rangle,$$

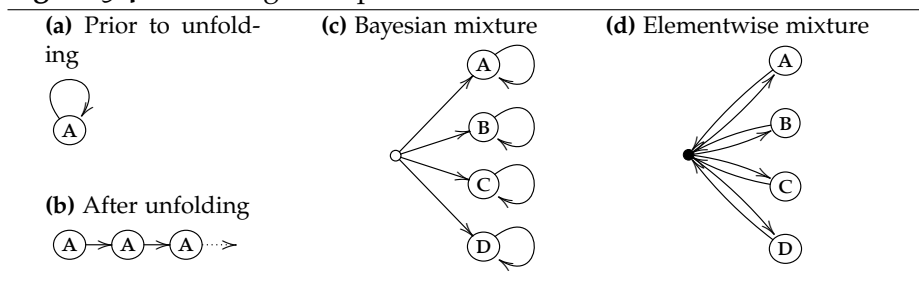
where the states and productive states are given by:

$$Q^u := \left\{ \langle q_\lambda, t \rangle \mid q^\lambda \text{ is a run through } \mathfrak{H} \right\}, \quad \text{where } t = |q_\lambda^\lambda|$$

$$Q_p^u := Q^u \cap (Q_p \times \mathbb{N})$$

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

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

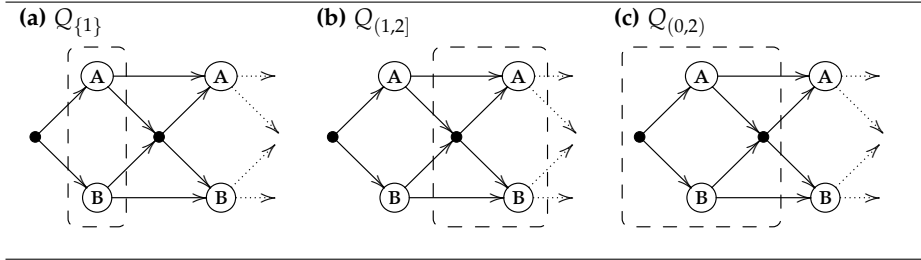
Figure 3.4 Unfolding example

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 the 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 $q, q' \in Q^u$. 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.

$$\begin{aligned}
 Q_{[s,t]}^u &:= Q^u \cap (Q \times [s,t]) & Q_{[s,t]}^u &:= Q_{[s,t]}^u \cup Q_{\{t\}}^u \\
 Q_{\{s\}}^u &:= Q^u \cap (Q_p \times \{s\}) & Q_{(s,t)}^u &:= Q_{[s,t]}^u \setminus Q_{\{s\}}^u \\
 & & Q_{(s,t]}^u &:= Q_{[s,t]}^u \setminus Q_{\{s\}}^u
 \end{aligned}$$

Fix $t > 0$, then $Q_{\{t\}}^u$ is a non-empty $<$ -anti-chain (i.e. its states are pairwise $<$ -incomparable). Furthermore $Q_{(t,t+1)}^u$ is empty if $Q_{\{t+1\}}^u = \bigcup_{q \in Q_{\{t\}}^u} S_q$, 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, $q \notin \text{dom}(w)$. Then at some point $w(q)$ is assigned the value $P_{\circ}(q)$. 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 $P_{\circ}(q) = 0$). 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) $\text{alg}(\mathfrak{H}, x^{t-1}, q)$. 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 $\text{alg}(\mathfrak{H}, x^t, q)$.

Algorithm 3.1 Forward algorithm. Fix an unfolded deterministic HMM prior $\mathfrak{H} = \langle Q, Q_p, P_o, P_r, \Lambda \rangle$ on Ξ , and an \mathcal{X} -PFS P_ξ for each expert $\xi \in \Xi$. The input consists of an infinite sequence x_1, x_2, \dots that arrives sequentially.

Declare the weight map (partial function) $w : Q \rightarrow [0, 1]$.
 $w(q) \leftarrow P_o(q)$ **for all** q s.t. $P_o(q) > 0$. $\triangleright \text{dom}(w) = I$
for $t = 1, 2, \dots$ **do**
 FORWARD PROPAGATION(t)
 Predict next expert:

$$P(\xi_t = \xi | x^{t-1}) = \frac{\sum_{q \in Q_{\{t\}} : \Lambda(q) = \xi} w(q)}{\sum_{q \in Q_{\{t\}}} w(q)}.$$

Predict next outcome:

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

LOSS UPDATE(t)

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

end for

FORWARD PROPAGATION(t)

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

LOSS UPDATE(t)

for all $q \in Q_{\{t\}}$ **do** $\triangleright q \in Q_p$
 $w(q) \leftarrow w(q) P_{\Lambda(q)}(x_t | x^{t-1})$.
end for

Note that we associate *two* weights with each productive state $q \in Q_{\{t\}}$: the weight $\text{alg}(\mathfrak{H}, x^{t-1}, q)$ is calculated *before* outcome t is observed, while $\text{alg}(\mathfrak{H}, x^t, q)$ 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 $\text{alg}(\mathfrak{H}, x^t, q) = P_{\mathfrak{H}}(x^t, q)$.*

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 $B_q = \{q' \mid P(q' \rightarrow q) > 0\}$ 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:

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

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:

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

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

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 $|\text{dom}(w)|$ many weights. But

at each sample size t , $\text{dom}(w) \subseteq Q_{[t,t+1]}$. 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

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

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 $O(kt)$ 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 $\{P_{\zeta^t} \mid \zeta^t \in \Xi^t\}$ 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 ζ^t will usually be expressed in terms of the parameter m , defined as the number of contiguous blocks in ζ^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 ,*

$$\ln \frac{P_{\xi^t}(x^t)}{P_{\pi}(x^t)} = \ln \frac{P_{\xi^t}(x^t)}{\sum_{\xi^t} P_{\xi^t}(x^t) \pi(\xi^t)} \leq \ln \frac{P_{\xi^t}(x^t)}{P_{\xi^t}(x^t) \pi(\xi^t)} = -\ln \pi(\xi^t). \quad (3.10)$$

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 $\text{BAYES}[\Xi, w]$ 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*

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

Proof. We rewrite

$$\begin{aligned} \frac{P_{\pi}(x^t)}{P_{\rho}(x^t)} &\geq \sum_{\xi^t: P_{\rho}(x^t, \xi^t) > 0} \frac{P_{\rho}(x^t, \xi^t)}{P_{\rho}(x^t)} \cdot \frac{P_{\pi}(x^t, \xi^t)}{P_{\rho}(x^t, \xi^t)} = \\ &\mathbb{E}_V \left[\frac{P_{\pi}(x^t, \xi^t)}{P_{\rho}(x^t, \xi^t)} \right] = \mathbb{E}_V \left[\frac{\pi(\xi^t)}{\rho(\xi^t)} \right], \end{aligned}$$

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

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 $T_{\eta}(j) = e^{\eta^{\top} \phi(j)} h(j) / Z(\eta)$ be an exponential family of distributions with parameter vector η , some sufficient statistic ϕ , carrier h

and normalisation $Z(\eta) = \sum_j e^{\eta^\top \phi(j)} h(j)$, where j takes values in a finite set \mathcal{J} . Define a successor function $S : Q^\dagger \times \mathcal{J} \rightarrow Q$. Now set $P(q \rightarrow q') = T_\eta(j)$, 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{S} 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 $W = P_{\hat{\eta}}(q^\lambda | x^t)$ denote the posterior of $P_{\hat{\eta}}$ on runs. We then have*

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

where $D(P \| Q) = \sum_x P(x) \log P(x) / Q(x)$ 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 $\mathbb{E}_W[n(q^\lambda)]$ 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

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

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

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

we obtain²

$$\begin{aligned}
\vec{0} &= -\nabla_\eta \frac{P_\eta(x^t)}{P_{\hat{\eta}}(x^t)} \Big|_{\eta=\hat{\eta}} = -\sum_{\mathbf{q}^\lambda} \frac{P_{\hat{\eta}}(x^t, \mathbf{q}^\lambda)}{P_{\hat{\eta}}(x^t)} \nabla_\eta \ln P_\eta(x^t, \mathbf{q}^\lambda) \Big|_{\eta=\hat{\eta}} = \\
&\qquad \qquad \qquad \nabla_\eta \mathbb{E}_W \left[\ln \frac{\pi_{\hat{\eta}}(\mathbf{q}^\lambda)}{\pi_\eta(\mathbf{q}^\lambda)} \right] \Big|_{\eta=\hat{\eta}}.
\end{aligned}$$

This shows that the vector differential of (3.11) must be zero at $\hat{\eta}$. Re-ordering terms we obtain

$$\begin{aligned}
\sum_j \phi(j) \mathbb{E}_W[n_j(\mathbf{q}^\lambda)] &= \mathbb{E}_W[n(\mathbf{q}^\lambda)] \nabla_\eta \ln \frac{Z(\eta)}{Z(\hat{\eta})} \Big|_{\eta=\hat{\eta}} = \\
&\qquad \qquad \qquad \mathbb{E}_W[n(\mathbf{q}^\lambda)] \mathbb{E}_{j \sim T_{\hat{\eta}}} [\phi(j)], \tag{3.12}
\end{aligned}$$

where the last step follows from

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

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

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

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

completing the proof. \square

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 $\phi(j) = (0, \dots, 1, 0, \dots)$, 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 $D(T_{\hat{\eta}} \| T_{\eta})$ 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 $\text{BAYES}[\Xi, w]$ 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 $\mathbb{E}_W[n(\mathbf{q}^\lambda)] = 1$, so the lemma tells us that our regret is $D(\hat{w} \| w)$, 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 $\hat{\zeta}$, so $D(\hat{w} \| w) = -\ln w(\hat{\zeta})$ as before. \diamond

3.4.5. EXAMPLE (Regret of Elementwise Mixtures). We now compute the regret of $\text{EM}[\Xi, w]$. Let Q^\dagger contain the silent states and again identify the experts Ξ with \mathcal{J} . For elementwise mixtures, $\mathbb{E}_W[n(\mathbf{q}^\lambda)] = t$. 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 $t D(\hat{w} \| w)$. \diamond

3.5 Switching Strategies

3.5.1 Fixed Share

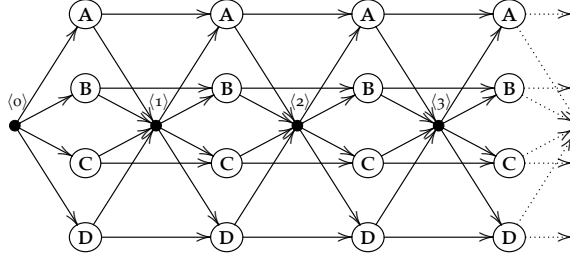
The paper by Herbster and Warmuth [79, 80] on *tracking the best expert* is the first to consider the scenario where the best predicting expert may change with the sample size. They do this by partitioning the data of size t into m segments, where each segment is associated with an expert, and give algorithms 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 $\text{FS}[\Xi, w, \alpha]$ 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 ζ^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 ζ^t . Write $H(\alpha^*, \alpha) = -\alpha^* \ln \alpha - (1 - \alpha^*) \ln(1 - \alpha)$ for the cross entropy. Then*

$$\ln \frac{P_{\zeta^t}(x^t)}{P_{\text{FS}[\Xi, w, \alpha]}(x^t)} \leq m \ln k + (t - 1) H(\alpha^*, \alpha). \quad (3.13)$$

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

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

$$\begin{aligned}
 Q &= Q_s \cup Q_p \quad Q_s = \mathbb{N} \quad Q_p = \Xi \times \mathbb{Z}_+ \\
 P_0(0) &= 1 \quad \Lambda(\xi, t) = \xi \\
 P \begin{pmatrix} \langle t \rangle \rightarrow \langle \xi, t+1 \rangle \\ \langle \xi, t \rangle \rightarrow \langle t \rangle \\ \langle \xi, t \rangle \rightarrow \langle \xi, t+1 \rangle \end{pmatrix} &= \begin{pmatrix} w(\xi) \\ \alpha \\ 1 - \alpha \end{pmatrix}
 \end{aligned}$$

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

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

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 $\hat{\alpha} = \arg\max_{\alpha} P_{\text{fs}[\Xi, w, \alpha]}(x^t)$. 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

$$\ln \frac{P_{\text{fs}[\Xi, w, \hat{\alpha}]}(x^t)}{P_{\text{fs}[\Xi, w, \alpha]}(x^t)} \leq (t - 1) D(\hat{\alpha} \| \alpha). \quad (3.14)$$

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

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

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}_n and \mathbb{B}_s . The *interpolator* \mathfrak{H} specifies *when* switches occur, \mathbb{B}_n determines the *normal* (n) evolution and \mathbb{B}_s determines the evolution when a *switch* (s) occurs. In particular, we obtain a model that defines the same distribution as $\text{FS}[\Xi, w, \alpha]$ by interpolation using $\mathfrak{H} = \text{EM}[\{n, s\}, (1 - \alpha, \alpha)]$, $\mathbb{B}_n = \text{BAYES}[\Xi, w]$ and $\mathbb{B}_s = \text{EM}[\Xi, w]$.

3.5.2. DEFINITION (Interpolation). See Figure 3.7 for an illustration. Let $\mathfrak{H} = \langle Q^{\mathfrak{H}}, Q_p^{\mathfrak{H}}, P_o^{\mathfrak{H}}, P_s^{\mathfrak{H}}, \Lambda^{\mathfrak{H}} \rangle$ be a deterministic unfolded HMM on $\{n, s\}$, and let \mathbb{B}_n and \mathbb{B}_s be deterministic unfolded HMMs on experts Ξ sharing a common state set Q with identical initial distribution P_o and interpretation Λ (and thus identical productive states Q_p). We define

$\mathbb{B}_n \otimes_{\mathfrak{H}} \mathbb{B}_s$, the \mathfrak{H} -interpolation of \mathbb{B}_n and \mathbb{B}_s , by

$$\mathbb{B}_n \otimes_{\mathfrak{H}} \mathbb{B}_s := \langle Q^\otimes, Q_p^\otimes, P_o^\otimes, P_\rightarrow^\otimes, \Lambda^\otimes \rangle.$$

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:

$$Q^\otimes := Q \times Q_{\mathfrak{H}} \cup Q_p \times Q_{\mathfrak{H}},$$

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

$$Q_p^\otimes := \bigcup_{t \geq 1} Q_{\{t\}} \times Q_{\mathfrak{H}}^{\{t\}}.$$

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

$$P_o^\otimes(\langle q, a \rangle) := P_o(q) P_o^{\mathfrak{H}}(a).$$

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

$$P_\rightarrow^\otimes(\langle q, a \rangle \rightarrow \langle q', a' \rangle) := \begin{cases} P_\rightarrow^{\mathbb{B}_a}(q \rightarrow q') & \text{if } q \in Q_{[t, t+1]}, a = a' \in Q_{\mathfrak{H}}^{\{t\}}, \\ P_\rightarrow^{\mathfrak{H}}(a \rightarrow a') & \text{if } a \in Q_{[t, t+1]}^{\mathfrak{H}}, q = q' \in Q_{\{t+1\}}, \\ 0 & \text{otherwise.} \end{cases}$$

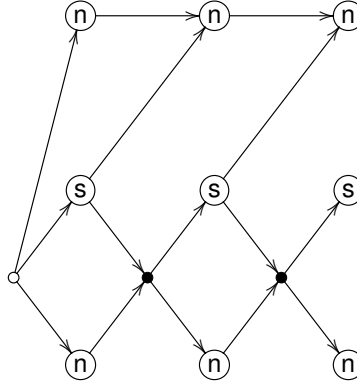
Finally, the node label is that of the first component

$$\Lambda^\otimes(\langle q, a \rangle) := \Lambda(q).$$

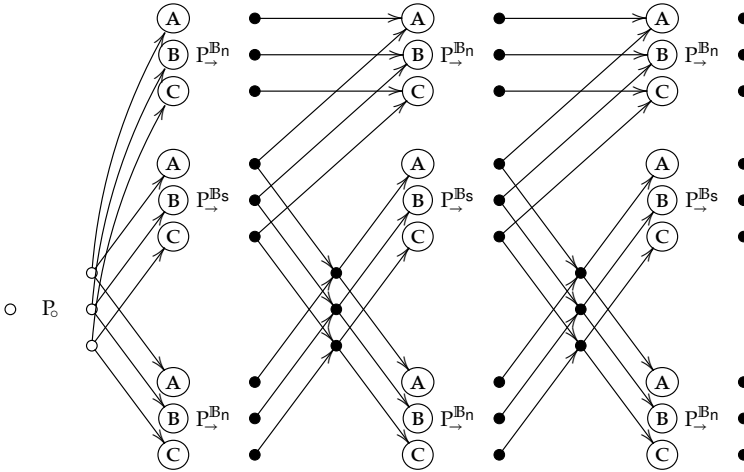
Figure 3.7b shows the state transition diagram of an interpolation, with the interpolator shown in Figure 3.7a. Figure 3.7c displays the Bayesian

Figure 3.7 Interpolation example: graph structure

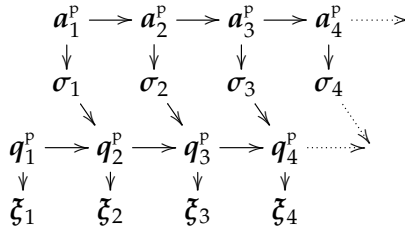
(a) HMM \mathcal{H} on $\{n, s\}$ (normal/switch)



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



(c) Bayesian network of interpolation



network of an interpolation. The random variables q_i^p and a_i^p are the components of the productive state at time i , while $\xi_i = \Lambda(q_i^p)$ and $\sigma_i = \Lambda^{\mathfrak{H}}(a_i^p)$. Note that $\sigma_i = \mathbf{s}$ 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}_n and \mathbb{B}_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 $\sigma^{t-1} \in \{\mathbf{n}, \mathbf{s}\}^{t-1}$ of switch decisions (on the \mathfrak{H} level) and each sequence $q_p^t \in Q_p^t$ of productive states (on the \mathbb{B} level)*

$$\pi_{\mathbb{B}_n \otimes_{\mathfrak{H}} \mathbb{B}_s}(q_p^t) \geq \pi_{\mathfrak{H}}(\sigma^{t-1}) P_\circ(q_1^p) \prod_{i=1}^{t-1} \pi_{\sigma_i}(q_{i+1}^p | q_i^p).$$

Proof. For every distribution

$$\pi_{\mathbb{B}_n \otimes_{\mathfrak{H}} \mathbb{B}_s}(q_p^t) \geq \pi_{\mathbb{B}_n \otimes_{\mathfrak{H}} \mathbb{B}_s}(\sigma^{t-1}) \pi_{\mathbb{B}_n \otimes_{\mathfrak{H}} \mathbb{B}_s}(q_p^t | \sigma^{t-1}).$$

By the definition of interpolation, we have $\pi_{\mathbb{B}_n \otimes_{\mathfrak{H}} \mathbb{B}_s}(\sigma^{t-1}) = \pi_{\mathfrak{H}}(\sigma^{t-1})$ and $\pi_{\mathbb{B}_n \otimes_{\mathfrak{H}} \mathbb{B}_s}(q_p^t | \sigma^{t-1}) = P_\circ(q_1^p) \prod_{i=1}^{t-1} \pi_{\sigma_i}(q_{i+1}^p | q_i^p)$. \square

3.5.4. COROLLARY (Default Interpolation Regret). *We apply this theorem to our \mathbb{B} -level HMMs of interest, $\mathbb{B}_n = \text{BAYES}[\Xi, w]$ and $\mathbb{B}_s = \text{EM}[\Xi, w]$ where w is uniform on k experts. Fix ζ^t . Set $\sigma_i = \mathbf{s}$ iff $\zeta_{i+1} \neq \zeta_i$, and let m be the number of blocks in ζ^t , i.e. the number of \mathbf{s} in σ^{t-1} plus one. Then for all data x^t*

$$\ln \frac{P_{\zeta^t}(x^t)}{P_{\mathbb{B}_n \otimes_{\mathfrak{H}} \mathbb{B}_s}(x^t)} \leq -\ln \pi_{\mathfrak{H}}(\sigma^{t-1}) + m \ln k.$$

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 $P_\circ(\zeta_1) = w(\zeta_1)$,

$$\pi_{\text{BAYES}[\Xi, w]}(\zeta_i = \zeta_{i-1} | \zeta_{i-1}) = 1, \quad \text{and} \quad \pi_{\text{EM}[\Xi, w]}(\zeta_i | \zeta_{i-1}) = w(\zeta_i).$$

Lemma 3.5.3, using $w(\zeta) = 1/k$, yields $\pi_{\mathbb{B}_n \otimes_{\mathfrak{H}} \mathbb{B}_s}(\zeta^t) \geq \pi_{\mathfrak{H}}(\sigma^{t-1}) k^{-m}$ and the result follows by (3.10). \square

3.5.5. EXAMPLE (Fixed Share Regret). We shorten $\text{EM}[\{n, \mathbf{s}\}, (1 - \alpha, \alpha)]$ to $\text{FS}[\alpha]$. We now redefine the fixed share model in terms of the interpolation

$$\text{FS}[\Xi, w, \alpha] := \text{BAYES}[\Xi, w] \otimes_{\text{FS}[\alpha]} \text{EM}[\Xi, w].$$

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

$$-\ln \pi_{\text{FS}[\alpha]}(\sigma^{t-1}) = -\ln((1 - \alpha)^{t-m} \alpha^{m-1}) = (t - 1)H(\alpha^*, \alpha). \quad (3.16)$$

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 $m \ln k$. \diamond

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 $\mathbb{B}_n \otimes_{\mathfrak{H}} \mathbb{B}_s$, we need to compute $|Q_{\{t\}}|$ many forward propagations on $\mathbb{B}_s/\mathbb{B}_n$, followed by $|Q_{\{t\}}^{\mathfrak{H}}|$ 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}_n and \mathbb{B}_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, $\text{DSR}[\alpha^\omega]$, which is similar to $\text{FS}[\alpha]$, 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

$$\text{DSR}[\Xi, w, \alpha^\omega] := \text{BAYES}[\Xi, w] \otimes_{\text{DSR}[\alpha^\omega]} \text{EM}[\Xi, w].$$

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

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

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 $\alpha_i = 1 - e^{-c/i}$ 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 ζ^t with m blocks*

$$\ln \frac{P_{\zeta^t}(x^t)}{P_{\text{DSR}[\Xi, w, \alpha^\omega]}(x^t)} \leq m \ln k + c - (m-1) \ln c + (m-1+c) \ln(t-1). \quad (3.18)$$

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

$$\begin{aligned} -\ln \pi_{\text{DSR}[\alpha^\omega]}(\sigma^{t-1}) &= c \sum_{i=1}^{t-1} \frac{1}{i} - \sum_{j=2}^m \ln(e^{c/t_j} - 1) \leq \\ & c \ln(t-1) + c - (m-1) \ln c + \sum_{j=2}^m \ln t_j. \end{aligned}$$

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

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 $\text{DSR}[\alpha^\omega]$ 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 $\alpha_i = 1 - e^{-c\tau(i)}$ 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 ζ^t with m blocks*

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

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

$$\begin{aligned} -\ln \pi_{\text{DSR}[\alpha^\omega]}(\sigma^{t-1}) &= c \sum_{i=1}^{t-1} \tau(i) - \sum_{j=2}^m \ln(e^{c \cdot \tau(t_j)} - 1) \leq \\ & c - (m - 1) \ln c - \sum_{j=2}^m \ln \tau(t_j). \end{aligned}$$

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. \square

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

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

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

$$\text{SM}[\Xi, w] := \text{BAYES}[\Xi, w] \otimes_{\text{SM}} \text{EM}[\Xi, w],$$

with the interpolator sm defined in Figure 3.8. Each productive state $\langle n_n, n_s, \sigma \rangle$ represents the fact that after observation $n_n + n_s + 1$ a switch occurs ($\sigma = \text{s}$) or not ($\sigma = \text{n}$), 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 $\{P_{\text{FS}[\Xi, w, \alpha]} \mid \alpha \in [0, 1]\}$. 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*

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

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)

$$\begin{aligned} \max_{x^t} \frac{P_{\text{FS}[\Xi, w, \alpha]}(x^t)}{P_{\text{SM}[\Xi, w]}(x^t)} &\leq \max_{\zeta^t} \frac{P_{\text{FS}[\Xi, w, \alpha]}(\zeta^t)}{P_{\text{SM}[\Xi, w]}(\zeta^t)} \leq \\ &\max_{\sigma^{t-1}} \frac{P_{\text{FS}[\Xi, w, \alpha]}(\sigma^{t-1})}{P_{\text{SM}[\Xi, w]}(\sigma^{t-1})} = \max_{\sigma^{t-1}} \frac{P_{\text{FS}[\alpha]}(\sigma^{t-1})}{P_{\text{SM}}(\sigma^{t-1})}. \end{aligned}$$

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 . \square

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

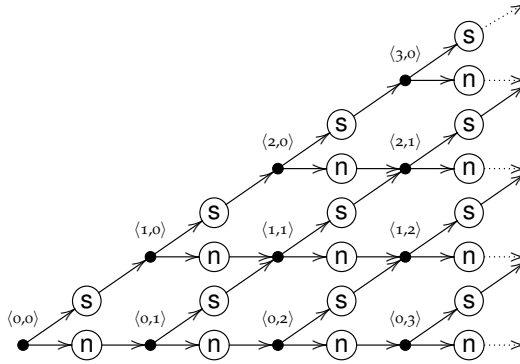
$$\ln \frac{P_{\zeta^t}(x^t)}{P_{\text{SM}[\Xi, w]}(x^t)} \leq m \ln k + (t-1) H(\alpha^*) + \ln 2 + \frac{1}{2} \ln t.$$

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



$$Q = Q_s \cup Q_p \quad Q_s = \mathbb{N}^2 \quad Q_p = \mathbb{N}^2 \times \{n, s\}$$

$$\Lambda(n_n, n_s, \sigma) = \sigma \quad P_o(0, 0) = 1$$

$$P_{\rightarrow} \begin{pmatrix} \langle n_n, n_s, n \rangle \rightarrow \langle n_n + 1, n_s \rangle \\ \langle n_n, n_s, s \rangle \rightarrow \langle n_n, n_s + 1 \rangle \\ \langle n_n, n_s \rangle \rightarrow \langle n_n, n_s, n \rangle \\ \langle n_n, n_s \rangle \rightarrow \langle n_n, n_s, s \rangle \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ \frac{(n_n + \frac{1}{2})}{(n_n + n_s + 1)} \\ \frac{(n_s + \frac{1}{2})}{(n_n + n_s + 1)} \end{pmatrix}$$

the number of states quadratic. The quadratic running time $O(kt^2)$ 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 $O(kt\sqrt{t})$.

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.

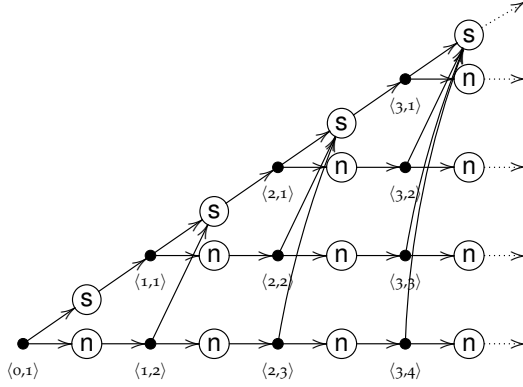
The run-length model is based on the assumption that the *intervals* between successive switches are independently distributed according to some distribution τ . After the universal share model and the switch distribution, this is a third generalisation of the fixed share algorithm, which is recovered 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 $\text{RL}[\tau]$ 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 δ th sample since the last switch. The HMM for the run-length model is given by the interpolation

$$\text{RL}[\Xi, w, \tau] := \text{BAYES}[\Xi, w] \otimes_{\text{RL}[\tau]} \text{EM}[\Xi, w].$$

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 $\text{RL}[\tau, c]$



$$Q = Q_s \cup Q_p \quad Q_s = \mathbf{S} \quad Q_p = \{\mathbf{n}\} \times \mathbf{S} \cup \{\mathbf{s}\} \times \mathbb{N}$$

$$P_0(0,1) = 1 \quad \Lambda(\mathbf{n}, t, \delta) = \mathbf{n} \quad \Lambda(\mathbf{s}, t) = \mathbf{s}$$

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

where

$$\mathbf{S} := \{ \langle t, \delta \rangle \in \mathbb{N}^2 \mid \delta \leq t+1 \}.$$

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 ζ^t with m blocks, we have*

$$\ln \frac{P_{\zeta^t}(x^t)}{P_{\text{RL}[\mathbb{E}, w, \tau]}(x^t)} \leq m \ln k - \ln \tau(\infty) - (m-1) \ln \tau \left(\frac{t_m}{m-1} \right). \quad (3.21)$$

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

above, we obtain

$$\begin{aligned} -\ln \pi_{\text{RL}[\tau]}(\sigma^{t-1}) &= -\ln \tau(\mathbf{z} \geq t - t_m) - \sum_{j=1}^{m-1} \ln \tau(\delta_j) \leq \\ &-\ln \tau(\infty) - \sum_{j=1}^{m-1} \ln \tau(\delta_j). \end{aligned}$$

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. \square

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

$$(\kappa * \lambda)(x) := \mathbb{E}_{\delta \sim \kappa} [\lambda(x - \delta)].$$

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}_s are replaced by a convolution κ on the experts. The HMM implementing these convolutions is $\text{KERNEL}[\kappa] := \langle Q, Q_p, P_o, P_r, \Lambda \rangle$, defined as follows

$$\begin{aligned} Q = Q_p &= \mathbb{Z} \times \mathbb{Z}_+ & \Lambda(\xi, t) &= \xi \\ P_o(\xi, 1) &= \kappa(\xi) & P(\langle \xi, t \rangle \rightarrow \langle \xi', t+1 \rangle) &= \kappa(\xi' - \xi). \end{aligned}$$

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

3.5.10. COROLLARY (Kernel Interpolation Regret). *Let $\mathbb{B}_n = \text{BAYES}[\mathbb{Z}, \kappa]$ and $\mathbb{B}_s = \text{KERNEL}[\kappa]$. Fix ξ^t . Set $\sigma_i = \mathbf{s}$ iff $\xi_{i+1} \neq \xi_i$, and for $1 \leq j \leq m$ let k_j denote the expert used in the j th block. Further let $k_0 = 0$. Then for all x^t :*

$$\ln \frac{P_{\xi^t}(x^t)}{P_{\mathbb{B}_n \otimes \mathbb{B}_s}(x^t)} \leq -\ln \pi_{\mathfrak{S}}(\sigma^{t-1}) - \sum_{j=1}^m \ln \kappa(k_j - k_{j-1}).$$

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

$$\begin{aligned} \pi_{\text{BAYES}[\mathbb{Z}, \kappa]}(\xi_1) &= \pi_{\text{KERNEL}[\kappa]}(\xi_1) = \kappa(\xi_1), \\ \pi_{\text{BAYES}[\mathbb{Z}, \kappa]}(\xi_i = \xi_{i-1} | \xi_{i-1}) &= 1, \end{aligned}$$

and

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

Now Lemma 3.5.3 yields $\pi_{\mathbb{B}_n \otimes \mathbb{B}_s}(\xi^t) \geq \pi_{\mathfrak{S}}(\sigma^{t-1}) \prod_{j=1}^m \kappa(k_j - k_{j-1})$, and the result follows by (3.10). \square

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 ζ^t :

$$d = \sum_{i=1}^t |\delta_i|, \quad \text{where} \quad \delta_1 = \zeta_1 \quad \text{and} \quad \delta_i = \zeta_i - \zeta_{i-1} \quad \text{for } 1 < i \leq t,$$

which can be viewed as the length of the path described by ζ^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 `KERNEL`[κ_α] from the previous section, but as it turns out it is possible to represent the same

kernel using a different HMM $\text{PD}[\alpha]$, 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*

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

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

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

$$-\ln \pi_{\text{PD}[\alpha]}(\xi^t) = -\ln \prod_{i=1}^t \kappa_\alpha(\delta_i) = t \mathbb{E}_{\kappa_{\alpha^*}} [-\ln \kappa_\alpha(\delta)] = t H(\kappa_{\alpha^*}, \kappa_\alpha).$$

The right equality follows from

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

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

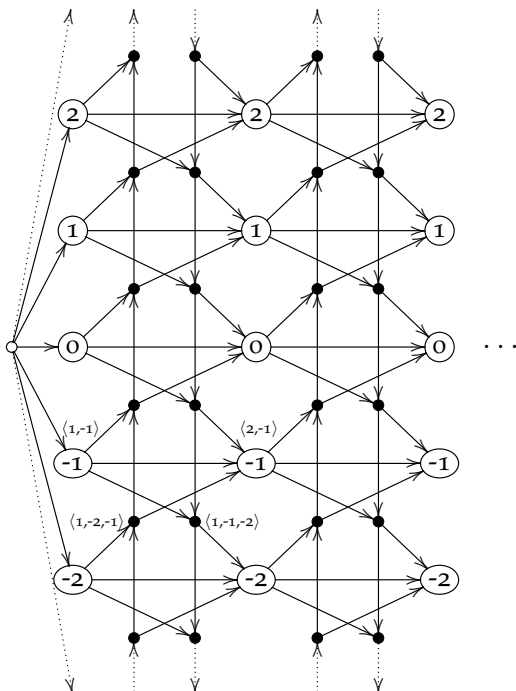
We can be somewhat more precise about how much it can hurt performance to use a suboptimal parameter α . The following theorem, which bounds the regret with respect to the optimal parameter-drift model, is an analogue of Equation 3.14 for fixed share. The theorem applies to a wide class of kernel HMMs,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

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

$$\ln \frac{P_{\text{KERNEL}[\kappa_{\hat{\eta}}]}(x^t)}{P_{\text{KERNEL}[\kappa_\eta]}(x^t)} \leq t D(\kappa_{\hat{\eta}} \parallel \kappa_\eta)$$

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^+ equal to the set of all productive states. \square

Figure 3.10 Parameter drift: $\text{PD}[\alpha]$



$$\begin{aligned}
 \text{PD}[\alpha] &= \langle Q, Q_p, P_o, P_s, \Lambda \rangle & Q &= Q_s \cup Q_p \\
 Q_p &= \mathbb{Z}_+ \times \mathbb{Z} & P_o(\langle 1, \xi \rangle) &= \kappa_\alpha(\xi) & \Lambda(t, \xi) &= \xi \\
 Q_s &= \mathbb{Z}_+ \times \{ \langle i, i+1 \rangle, \langle i, i-1 \rangle \mid i \in \mathbb{Z} \} \\
 P & \begin{pmatrix} \langle t, \xi-1, \xi \rangle \rightarrow \langle t, \xi, \xi+1 \rangle \\ \langle t, \xi+1, \xi \rangle \rightarrow \langle t, \xi, \xi-1 \rangle \\ \langle t, \xi-1, \xi \rangle \rightarrow \langle t+1, \xi \rangle \\ \langle t, \xi+1, \xi \rangle \rightarrow \langle t+1, \xi \rangle \\ \langle t, \xi \rangle \rightarrow \langle t+1, \xi \rangle \\ \langle t, \xi \rangle \rightarrow \langle t, \xi, \xi+1 \rangle \\ \langle t, \xi \rangle \rightarrow \langle t, \xi, \xi-1 \rangle \end{pmatrix} = \begin{pmatrix} \alpha \\ \alpha \\ 1-\alpha \\ 1-\alpha \\ (1-\alpha)/(1+\alpha) \\ \alpha/(1+\alpha) \\ \alpha/(1+\alpha) \end{pmatrix}
 \end{aligned}$$

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 ζ^t with total drift d , we have*

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

Proof. We first expand

$$\begin{aligned} \pi_{\text{PD}}(\zeta^t) &= \prod_{i=1}^t \kappa_{\alpha_i}(\delta_i) = \prod_{i=1}^t \alpha_i^{|\delta_i|} \frac{1-\alpha_i}{1+\alpha_i} = \prod_{i=1}^t (i+1)^{-|\delta_i|} \frac{i}{i+2} = \\ &= \frac{2}{(t+1)(t+2)} \prod_{i=1}^t (i+1)^{-|\delta_i|}. \end{aligned}$$

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

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

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

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 $\text{FS}[\alpha]$ 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

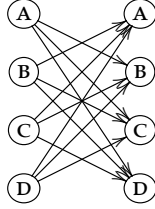
$$P(x^t) = \sum_{\zeta^t} P(x^t, \zeta^t),$$

and wonder about the sequence of *experts* ζ^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 ζ^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 $P(x^i, q_i^p)$ and $P(x^t | q_i^p, x^i)$. Recall that q_i^p is the productive state that is used at time i . From these we can compute the a posteriori probability $P(q_i^p | x^t)$ 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 $P(\zeta_i | x^t)$ of each *expert* $\zeta_i \in \Xi$ at each time $i = 1, \dots, t$. 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

$$\begin{aligned} P(\zeta \rightarrow \zeta') &= w(\zeta' | \zeta' \neq \zeta) \\ &= \frac{w(\zeta')}{1 - w(\zeta)} \end{aligned}$$

(c) Linear-time implementation

Input: Weights a_1, \dots, a_k at time t **Output:** Weights b_1, \dots, b_k at time $t + 1$

$$p \leftarrow \sum_{i=1}^k \frac{a_i}{1 - w(i)}$$

for $i = 1 \dots k$ **do**

$$b_i \leftarrow w(i) \left(p - \frac{a_i}{1 - w(i)} \right)$$

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}_s instead of $\text{EM}[\Xi, w]$. 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.