# Freezing and Sleeping: Tracking Experts that Learn by Evolving Past Posteriors

**Wouter M. Koolen** and **Tim van Erven***
Centrum Wiskunde & Informatica (CWI)
Science Park 123, P.O. Box 94079
1090 GB Amsterdam, The Netherlands
{Wouter.Koolen,Tim.van.Erven}@cwi.nl



## Abstract

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

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

## 1 Introduction

Freund's problem arises in the context of prediction with expert advice [2]. In this setting a sequence of outcomes needs to be predicted, one outcome at a time. Thus, prediction proceeds in rounds: in each round we first consult a set of experts, who give us their predictions. Then we make our own prediction and incur some loss based on the discrepancy between this prediction and the actual outcome. The goal is to minimise the difference between our cumulative loss and some reference scheme. For this reference there are several options; we may, for example, compare ourselves to the cumulative loss of the best expert in hindsight. A more ambitious reference scheme was proposed by Yoav Freund in 2000.

*We would like to thank Manfred Warmuth for raising our interest in this subject during COLT 2008. We also thank Steven de Rooij and Peter Grünwald for fruitful discussions and suggestions. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

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

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

In 2001, Freund's problem was addressed by Bousquet and Warmuth, who developed the efficient mixing past posteriors (MPP) algorithm [1]. MPP's loss is bounded by the loss of Freund's scheme plus some overhead that depends on the number of bits required to encode the partition of the data, and it has found successful application in [5]. Problem solved. Or is it?

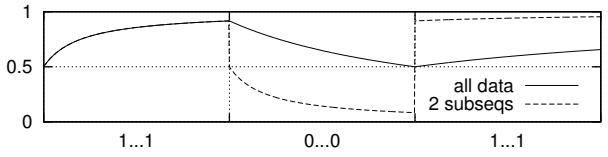
**The Loss of an Expert on a Subsequence** In our view Freund's problem has two possible interpretations, which differ most clearly for learning experts. Namely, to measure the predictive performance of an expert on a subsequence, do we show her the data *outside* her subsequence or not? An expert that sees all outcomes will track the *global* properties of the data. This is (implicitly) the case for mixing past posteriors. But an expert that only observes the subsequence that she has to predict might see and thus exploit its *local* structure, resulting in decreased loss. The more the characteristics of the subsequences differ, the greater the gain. Let us illustrate this by an example.

**Ambiguity Example** The data consist of a block of ones, followed by a block of zeros, again followed by a block of ones. In the reference scheme step (a), we split the data into two subsequences, one consisting of only ones, the other consisting of only zeros. Our expert predicts the probability of a one using Laplace's rule of succession, i.e. she *learns* the frequency of ones in the data that she observes [2]. Note that one learning expert suffices, as we can select her for both subsequences.

First we consider the expert's predictions when she observes all data. During the first block, our expert will increase her probability of a one from $1/2$ to nearly one. Then during the second block it will go down to $1/2$ again. During the third block it will increase from $1/2$ up, but slower. Thus, for block two the expert is extremely bad, while for block three she is at best mediocre. (See Figure 1.)

Compare this to the expert's predictions on the subsequences. During the subsequence of ones (first and third block), our

**Figure 1** Estimated probability of a one




expert will increase her probability of a one from $1/2$ to almost one, while during the subsequence of zeroes she will decrease her probability from $1/2$ to nearly zero. Thus, the expert is much better on the subsequences in isolation.

This shows that the predictive performance of a learning expert on a subsequence in isolation can be dramatically higher than that on the same sequence in the context of all data. This behaviour is typical: on all data a learning expert will learn the average, global pattern, while on a well-chosen subsequence she can zoom in on the local structure.

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

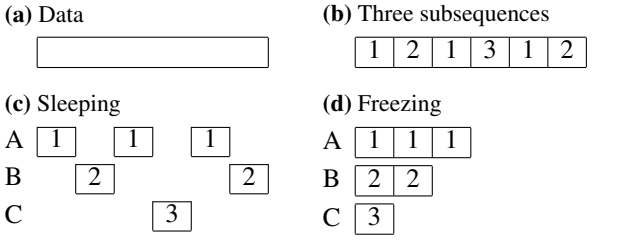
Often however, experts have internal structure. For example, in [11, 8, 15, 16] adaptive prediction strategies (i.e., learning experts) are explicitly constructed from basic experts. To represent such structured experts, we use a general framework called *expert hidden Markov models* (EHMMs), that was introduced in [9]. EHMMs are hidden Markov models in which the production probabilities are determined by expert advice. A structured expert in EHMM form provides sufficient information about its predictions on any isolated subsequence.

Many strategies for prediction with expert advice (i.e. learning experts) can be rendered as EHMMs. For example all adaptive strategies in the papers above (see [9]). But there are also strategies that cannot be brought into EHMM form, like e.g. *follow the perturbed leader* [6] and *variable share* [8].

Our approach may also be of interest to machine learning with regular hidden Markov models (HMMs) [12]. Although existing approaches to shift between multiple HMMs [3, 4, 10] usually focus on change-point detection, prediction seems a highly related issue.

**Sleeping or Freezing** We evaluate the performance of learning experts on subsequences in isolation. But now another choice presents itself (see Figure 2). Should we present the subsequence to the expert consecutively (we view this as *freezing* the expert's state on other data)? Or should we retain the original timing of the selected samples and keep the intermediate samples unobserved (then the expert is *sleeping* for other data)? To illustrate the difference, consider an expert that is able to predict the television images of our favourite show. We want to *freeze* her during commercial breaks, so

**Figure 2** Freezing (consecutive) and Sleeping (timing preserved). Both use experts A,B,C on subsequences 1,2,3.




that she continues predicting the show where it left off. We want to put her to *sleep* when we zap to another channel, so that after zapping back, she will predict the show as it has advanced. Thus freezing vs sleeping is a modelling decision, that should be made on a case-by-case basis. We cover both scenarios.

### 1.1 Overview

After preliminaries we start by reviewing the main existing loss bound for mixing past posteriors in §3. Then, in §4, we introduce EHMMs as a way to represent structured experts.

The next section, §5, contains our results for Freund's problem when structured experts are evaluated on isolated subsequences. We formalise sleeping and freezing as two different ways of presenting a subsequence of the data to an EHMM, and present the *evolving past posteriors* EPP algorithm that takes an EHMM as input. The EPP algorithm has two variants, which both generalise the mixing past posteriors algorithm in a different way: EPP-SLEEPING for sleeping and EPP-FREEZING for freezing. The relation between EPP and other existing prediction strategies is shown in Figure 3. There $A \to B$ means that by carefully choosing prediction strategy $A$'s parameters it reduces to strategy $B$.

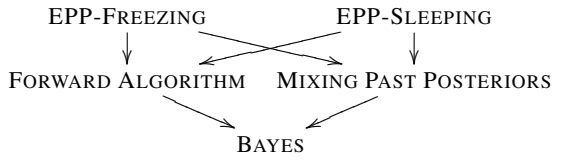
In order to understand EPP, we verify that it produces the same predictions for any two EHMMs that are equivalent in an appropriate sense, and analyse its running time. We then proceed to show our main result, which is that the losses of EPP-FREEZING and EPP-SLEEPING are bounded by the loss of Freund's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

We first derive our results only for logarithmic loss. This allows us to use familiar concepts and results from probability theory and refer to the interpretation of log loss as a codelength [2]. In §6 we conclude by proving that any algorithm that satisfies certain weak conditions, in particular EPP, directly generalises to an algorithm for arbitrary mixable losses with the appropriate loss bounds.

## 2 Preliminaries

**Prediction With Expert Advice** Each round $t$, we first receive advice from each expert $e \in \mathcal{E}$ in the form of an action

**Figure 3** Generalisation relation among prediction strategies




$a_t^e \in \mathcal{A}$. Then we distill our own action $a_t^{\mathsf{alg}} \in \mathcal{A}$ from the expert advice. Finally, the actual outcome $x_t \in \mathcal{X}$ is observed, and everybody suffers loss as specified by a fixed loss function $\ell\colon \mathcal{A} \times \mathcal{X} \to [0, \infty]$. Thus, the performance of a sequence of actions $a_1 \cdots a_T$ upon data $x_1 \cdots x_T$ is measured by the cumulative loss $\sum_{t=1}^{T} \ell(a_t, x_t)$.

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

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

## 3 Mixing Past Posteriors

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

We will not formally introduce MPP here, but recover it as a special case of both the freezing and sleeping algorithms in §5.4. Here we state the classical loss bound [1, Theorem 7], introducing our notation along the way. This loss bound relates the loss of MPP to Freund's scheme, where we choose a partition of the data (step a) and select an expert for each partition cell (step b). We measure expert performance (step c) using the predictions issued in the context of all data, i.e. the traditional interpretation of Freund's scheme.

### 3.1 Loss Bound

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

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

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

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

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

Note that this distribution is potentially *defective*; two elements $i < j$ cannot share the same nonzero predecessor, but $\beta_i$ may assign nonzero probability to $\mathrm{prev}^{\mathbb{C}}(j)$ nonetheless.

Now that we have seen how the loss bound encodes partition, we turn to $\mathrm{P}_{\mathbb{C}}^{\mathsf{B}}(x_{1:T})$, the probability of the data $x_{1:T}$ given a particular partition $\mathbb{C}$. To compute it, we treat the cells independently (2), and per cell we use the Bayesian mixture with prior $w$ on experts (3), thus mixing the predictions the experts issued in the context of all data (4).

$$\mathrm{P}_{\mathbb{C}}^{\mathsf{B}}(x_{1:T}) \coloneqq \prod_{\mathcal{C} \in \mathbb{C}} \mathrm{P}_{\mathbb{C}}^{\mathsf{B}}(x_{\mathcal{C}}), \text{ where} \tag{2}$$

$$\mathrm{P}_{\mathbb{C}}^{\mathsf{B}}(x_{\mathcal{C}}) \coloneqq \sum_{e \in \mathcal{E}} w(e)\mathrm{p}_{\mathcal{C}}^e(x_{\mathcal{C}}) \text{ and} \tag{3}$$

$$\mathrm{p}_{\mathcal{C}}^e(x_{\mathcal{C}}) \coloneqq \prod_{i \in \mathcal{C}} \mathrm{p}_i^e(x_i). \tag{4}$$

A second bounding step allows us to relate the performance of MPP directly to Freund's scheme. Let $w$ be the uniform prior over a finite set of experts $\mathcal{E}$, and select an expert $e^{\mathcal{C}}$ for each partition cell $\mathcal{C} \in \mathbb{C}$. Then bound each sum (3) from below by one of its terms to obtain

**Corollary 1.** $$\mathrm{P}_w^{\mathsf{MPP}}(x_{1:T}) \;\geq\; \beta(\mathbb{C})\,|\mathcal{E}|^{-|\mathbb{C}|} \prod_{\mathcal{C} \in \mathbb{C}} \mathrm{p}_{\mathcal{C}}^{e^{\mathcal{C}}}(x_{\mathcal{C}}).$$

Thus the log-loss overhead of MPP over Freund's scheme is bounded by $-\log \beta(\mathbb{C}) + |\mathbb{C}| \log|\mathcal{E}|$, which can be related to the number of bits to encode the chosen partition and the selected experts for each cell [1].

**Convex Combinations** In [1], the authors make a point of selecting a *convex combination of experts* for each subsequence, where the loss of a convex combination of experts is the weighted average *loss* of the experts. The loss of such a convex combination is therefore *always* higher than the loss of its best expert. Uniform bounds in terms of arbitrary experts, like Corollary 1, apply in particular to the best expert, and hence to any convex combination. Therefore, without loss of generality, we do not discuss convex combinations any further.

**Figure 4** Bayesian Network specification of an EHMM

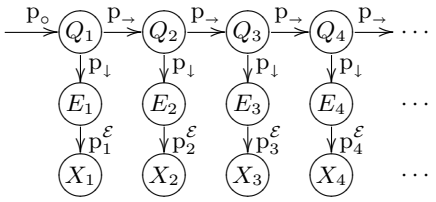
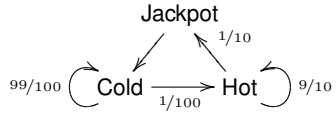



**Figure 5** Hidden state transitions in slot machine HMM




**Interpreting Freund's Problem** This loss bound shows that MPP solves the black-box-experts interpretation of Freund's problem. This can be seen clearly in (4). To predict the subsequence $x_{\mathcal{C}}$, it uses predictions $\mathrm{p}^e_{\mathcal{C}}$ which were issued in the context of all data. This means that the experts observe the entire history $x_{1:i}$ before predicting the next outcome $X_{i+1}$.

Switching between *learning* experts that observe all data is useful when the data are homogeneous, and the experts learn its global pattern at different speeds. In such cases we want to train each expert on all observations, for then by switching at the right time, we can predict each outcome using the expert that has learned most *until then*. This scenario is analysed in [14], where experts are parameter estimators for a series of statistical models of increasing complexity.

On the other hand, if the data have local patterns then our new interpretation of Freund's problem applies, and we want to train each expert on the subsequence on which it is evaluated, so that it can exploit its local patterns. To solve Freund's problem for such learning experts, we need to know about its internal structure.

# 4 Structured Experts

Assume there is only a single expert and fix a reference partition. Suppose we want to predict as if the expert is restarted on each cell of the partition, when in reality the expert just makes her predictions as if all the data were in a single cell. Then clearly this is impossible if we treat this expert completely as a black box: if we do not know what the expert's predictions would have been if a certain outcome were, say, the start of a new cell, then we cannot match these predictions.

The expert therefore needs to reveal to us some of her internal state. To this end, we will represent the parts of her internal state that will *not* be revealed to us by lower level experts that we will treat as black boxes, and assume our main expert combines the predictions of these base experts using an *expert hidden Markov model* (EHMM).

## 4.1 EHMMs

Expert Hidden Markov Models (EHMMs) were introduced in [9] as a language to specify strategies for prediction with expert advice. We briefly review them here. An *EHMM* $\mathfrak{H}$ is a probability distribution that is constructed according to the Bayesian network in Figure 4. It is used to sequentially predict outcomes $X_1, X_2, \ldots$ which take values in outcome space $\mathcal{X}$. At each time $t$, the distribution of $X_t$ depends on a hidden state $Q_t$, which determines mixing weights for the experts' predictions. Formally, the *production function* $\mathrm{p}_\downarrow$ determines the interpretation of a state: it maps any state $q_t \in \mathcal{Q}$ to a distribution $\mathrm{p}_\downarrow^{q_t}$ on the identity $E_t$ of the expert that should be used to predict $X_t$. Then given $E_t = e$, the distribution of $X_t$ is base expert $e$'s prediction $\mathrm{p}^e_t$. It remains to define the distribution of the hidden states. The starting state $Q_1$ has *initial distribution* $\mathrm{p}_\circ$, and the state evolves according to the *transition function* $\mathrm{p}_\rightarrow$, which maps any state $q_t$ to a distribution $\mathrm{p}_\rightarrow^{q_t}$ on states.

An EHMM $\mathfrak{H}$ defines a prediction strategy as follows; after observing $x_{1:t}$, predict outcome $X_{t+1}$ using the marginal $\mathfrak{H}(X_{t+1}|x_{1:t})$, which is a *mixture* of the expert's predictions $\mathrm{p}^{\mathcal{E}}_{t+1}$.

**Example 4.1** (Any Ordinary HMM)**.** To illustrate how ordinary HMMs are a special case of EHMMs, consider the following naive gambler's HMM model of an old-fashioned slot machine: in each round the gambler inserts one nickel into the slot machine and then the machine pays out a certain number of nickels depending on its hidden internal state: in state Cold it pays out nothing; in state Hot it pays out an amount between one and five nickels, uniformly at random; and then there's Jackpot in which it always pays out ten nickels. The machine always starts in state Cold and the state transitions are as in Figure 5.

To make an EHMM out of this HMM, we just identify experts with states: $\mathcal{Q} = \mathcal{E} = \{\mathsf{Cold}, \mathsf{Hot}, \mathsf{Jackpot}\}$, $\mathrm{p}^e_\downarrow(e) = 1$, and each expert predicts according to the corresponding payout scheme. The distributions on states follow the original HMM: $\mathrm{p}_\circ(\mathsf{Cold}) = 1$ and $\mathrm{p}_\rightarrow$ as in Figure 5. $\diamond$

**Example 4.2** (Bayes on base experts)**.** We identify the Bayesian distribution with prior $w$ on base experts $\mathcal{E}$ and the EHMM with $\mathcal{Q} = \mathcal{E}$, $\mathrm{p}_\circ = w$, and $\mathrm{p}^e_\rightarrow(e) = \mathrm{p}^e_\downarrow(e) = 1$, since their marginals coincide. Despite its deceptive simplicity, this EHMM *learns*: its marginal distribution on the next outcome is a mixture of the expert's predictions according to the Bayesian posterior. $\diamond$

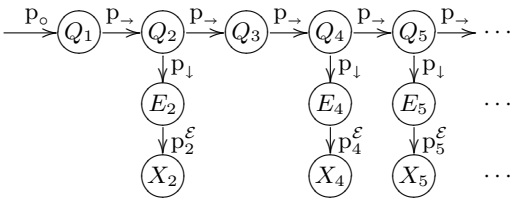
**Example 4.3** (Bayes on EHMMs)**.** Fix EHMMs $\mathfrak{H}^1, \ldots, \mathfrak{H}^n$ with disjoint state spaces and the same basic experts, and let $w$ be a prior distribution on $1{:}n$. The Bayesian mixture EHMM has state space $\mathcal{Q} = \bigcup_i \mathcal{Q}^i$, and for any two states $q, q' \in \mathcal{Q}^i$ belonging to the same original EHMM, $\mathrm{p}_\circ(q) = w(i)\,\mathrm{p}^i_\circ(q)$, $\mathrm{p}^q_\rightarrow(q') = \mathrm{p}^{i,q}_\rightarrow(q')$ and $\mathrm{p}^q_\downarrow(e) = \mathrm{p}^{i,q}_\downarrow(e)$. Again, this EHMM *learns* which of the given EHMMs is the best predictor. $\diamond$

## 4.2 The Forward Algorithm

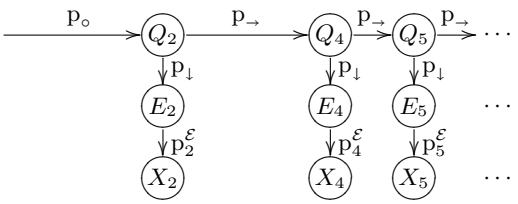
Sequential predictions for EHMMs can be computed efficiently using the *forward algorithm*, which maintains a posterior distribution over states, and predicts each outcome with a mixture of the experts' predictions [9]. Given a posterior $\lambda_t(Q_t) = \mathfrak{H}(Q_t|x_{1:t-1})$ for the hidden state at time $t$, the forward algorithm predicts $x_t$ using the marginal of $\mathfrak{H}(Q_t, E_t, X_t|x_{1:t-1})$.

**Figure 6** Sleeping and Freezing on outcomes $x_{\{2,4,5,\ldots\}}$

**(a)** Sleeping: EHMM $\mathfrak{H}^{\mathsf{sl}}_{\{2,4,5,\ldots\}}$




**(b)** Freezing: EHMM $\mathfrak{H}^{\mathsf{fr}}_{\{2,4,5,\ldots\}}$




Then, after observing outcome $x_t$, it updates its posterior $\lambda_t$ for $Q_t$ to a posterior $\lambda_{t+1}$ for $Q_{t+1}$.

For finite $\mathcal{Q}$, $\mathcal{E}$ and $\mathcal{X}$, the running time of the algorithm is determined by this last posterior update step, which in general may require $O\big(|\mathcal{Q}|^2\big)$ computation steps for each round $t$. On $T$ outcomes, this gives a total running time of $O\big(|\mathcal{Q}|^2 \cdot T\big)$. In Appendix A we provide a more careful analysis.

# 5 Freezing & Sleeping

Let $x_{1:T} = x_1, \ldots, x_T$ be a sequence of data and suppose that a reference partition $\mathbb{C}$ of $1{:}T$ is given in advance. We are interested in the performance of a structured expert $\mathfrak{H}_{\mathbb{C}}$, which for each cell $\mathcal{C} \in \mathbb{C}$ runs a separate instance of the structured expert $\mathfrak{H}$ on the subsequence $x_{\mathcal{C}}$. This leaves unspecified, however, whether the original timing of $x_{\mathcal{C}}$ should be preserved when $x_{\mathcal{C}}$ is presented to $\mathfrak{H}$. This is a modelling choice, which depends on the application at hand. We therefore treat both the case where the timing is preserved, which we call *sleeping*, and the case where the timing is not preserved, which we call *freezing*. (See also Figure 2 in the introduction.)

**Sleeping** We say that the instance of $\mathfrak{H}$ that is used to predict cell $\mathcal{C}$ is sleeping if it does notice the passing of time during outcomes outside of $\mathcal{C}$, even though it does not observe them. We write $\mathfrak{H}^{\mathsf{sl}}_{\mathcal{C}}$ for the resulting EHMM, which is shown in Figure 6a for the example $\mathcal{C} = \{2, 4, 5, \ldots\}$. Notice that $\mathfrak{H}^{\mathsf{sl}}_{\mathcal{C}}$ contains all five states $Q_{1:5}$, even though it does not observe $x_1$ or $x_3$. This has the effect that state transitions from e.g. $Q_2$ to $Q_4$ are composed of two transition steps according to $\mathrm{p}_\rightarrow$. The distributions on individual cells combine into the following distribution on all data $x_{1:T}$:

$$\mathfrak{H}^{\mathsf{sl}}_{\mathbb{C}}(x_{1:T}) \coloneqq \prod_{\mathcal{C}\in\mathbb{C}} \mathfrak{H}^{\mathsf{sl}}_{\mathcal{C}}(x_{\mathcal{C}}).$$

To memorize the nature of sleeping, one may think of the way television channels get interleaved as you zap between them: a channel not being watched is not paused, but instead continues broadcasting even when its content is not observed.

**Freezing** In freezing, the instance of $\mathfrak{H}$ that is used to predict cell $\mathcal{C} \in \mathbb{C}$ is frozen when outcomes outside of $\mathcal{C}$ occur: its internal state should not change based on those outcomes. (Of course we have no control over the base experts on which $\mathfrak{H}$ is based, so they may do whatever they please with such data. We therefore do have to preserve the timing of the base experts' predictions.) The resulting EHMM $\mathfrak{H}^{\mathsf{fr}}_{\mathcal{C}}$ is shown for the example $\mathcal{C} = \{2, 4, 5, \ldots\}$ in Figure 6b. Note that $Q_2$, $Q_4$ and $Q_5$ are the first, second and third state of $\mathfrak{H}^{\mathsf{fr}}_{\mathcal{C}}$; state transitions between them consist of a single transition step according to $\mathrm{p}_\rightarrow$. The resulting distribution on all data is defined by

$$\mathfrak{H}^{\mathsf{fr}}_{\mathbb{C}}(x_{1:T}) \coloneqq \prod_{\mathcal{C}\in\mathbb{C}} \mathfrak{H}^{\mathsf{fr}}_{\mathcal{C}}(x_{\mathcal{C}}).$$

One might associate freezing with the way different e-mail conversations get interleaved in your inbox (if it is sorted by order of message arrival): a conversation about your latest research is paused (remains frozen) regardless of how much spam you receive in between.

## 5.1 An Infeasible Solution

The freezing or sleeping distributions can be computed if the reference partition $\mathbb{C}$ is given in advance. The problem we are addressing, however, is that we do not assume $\mathbb{C}$ to be known. An easy (but impractical) solution to this problem is to predict according to the Bayesian mixture of all possible partitions: let $w$ be a prior on the set of all possible partitions and predict such that the joint distribution on all data is given by

$$\mathfrak{B}(x) \coloneqq \sum_{\mathbb{C}} w(\mathbb{C}) \mathfrak{H}^{\mathsf{f/s}}_{\mathbb{C}}(x),$$

where $\mathsf{f/s}$ denotes either $\mathsf{fr}$ for freezing or $\mathsf{sl}$ for sleeping. Lower bounding the sum by the term for the reference partition $\mathbb{C}$ directly gives an upper bound on the log loss:

$$-\log \mathfrak{B}(x) \leq -\log w(\mathbb{C}) - \log \mathfrak{H}^{\mathsf{f/s}}_{\mathbb{C}}(x).$$

To predict according to $\mathfrak{B}$ in general would require an exponential amount of state to keep track of all possible partitions, which is completely impractical. In the following section we therefore present generalisations to both sleeping and freezing of the mixing past posteriors algorithm and show that their running time is comparable to that of the forward algorithm on $\mathfrak{H}$ itself. Then in section §5.3 we prove bounds that relate the additional loss to the encoding cost of the reference partition $\mathbb{C}$.

## 5.2 The EPP Algorithm

Here we present a generalisation of the mixing past posteriors (MPP) algorithm, which we call *evolving past posteriors* (EPP). It is based on the view that MPP internally uses the Bayesian mixture of base experts, which is a standard EHMM. Given this perspective and after making the distinction between sleeping and freezing, the generalisation to other EHMMs is straightforward. We will discuss the connections between MPP and EPP in more detail in §5.4.

**Algorithm 1** EPP: Evolving Past Posteriors

**Input:**

- An EHMM $\mathfrak{H}$ with components $\mathrm{p}_\circ$, $\mathrm{p}_\rightarrow$ and $\mathrm{p}_\downarrow$ (see §4)
- A mixing scheme $\beta_1, \beta_2, \ldots$ (see §3.1 and §5.2.2)
- Expert predictions $\mathrm{p}^{\mathcal{E}}_1, \mathrm{p}^{\mathcal{E}}_2, \ldots$ and data $x_1, x_2, \ldots$

**Output:** Predictions $\mathrm{p}^{\mathsf{alg}}_1, \mathrm{p}^{\mathsf{alg}}_2, \ldots$
**Storage:** Past posteriors $\pi_1, \pi_2, \ldots$ on $\mathcal{Q}$, the states of $\mathfrak{H}$

**Algorithm**

1: Set the first posterior to the initial distribution of $\mathfrak{H}$
$$\pi_1(q_1) \;\leftarrow \mathrm{p}_\circ(q_1)$$
2: **for** $t = 1, 2, \ldots$ **do**
3: Form $\lambda_t$, the current configuration, as the $\beta_t$-mixture of past posteriors:
$$\lambda_t(q_t) \leftarrow \sum_{0 \le j < t} \beta_t(j)\pi_{j+1}(q_t).$$
4: Compute $\mathrm{p}^{\mathsf{alg}}_t$, the joint distribution on states, experts and outcomes:
$$\mathrm{p}^{\mathsf{alg}}_t(q_t, e_t, x_t) \leftarrow \lambda_t(q_t)\, \mathrm{p}^{q_t}_\downarrow(e_t)\mathrm{p}^{e_t}_t(x_t).$$
5: Predict $x_t$ using the marginal $\mathrm{p}^{\mathsf{alg}}_t(x_t)$,
6: Observe $x_t$. Suffer log loss
$$\ell^{\mathsf{alg}}_t \leftarrow -\log\bigl(\mathrm{p}^{\mathsf{alg}}_t(x_t)\bigr).$$
7: Perform loss update and state evolution to obtain the next posterior
$$\pi_{t+1}(q_{t+1}) \leftarrow \sum_{q_t \in \mathcal{Q}} \mathrm{p}^{\mathsf{alg}}_t(q_t|x_t)\, \mathrm{p}^{q_t}_\rightarrow(q_{t+1}).$$
8: Only for sleeping: perform state evolution for all past posteriors ($1 \le j \le t$)
$$\pi_j(q_{t+1}) \leftarrow \sum_{q_t \in \mathcal{Q}} \pi_j(q_t)\, \mathrm{p}^{q_t}_\rightarrow(q_{t+1}).$$
9: **end for**

The EPP algorithm has variants for sleeping and freezing, which are both given in Algorithm 1. It takes an EHMM $\mathfrak{H}$ and mixing scheme $\beta$ (see §3.1) as input. Given a distribution $\lambda_t$ on the hidden state $Q_t$ at time $t$, the EPP algorithm predicts $X_t$ exactly like the forward algorithm. It differs from the forward algorithm, however, in the way it computes $\lambda_t$. Whereas in the forward algorithm $\lambda_t$ may be interpreted as the posterior distribution on $Q_t$, in the EPP algorithm $\lambda_t$ is a $\beta$-mixture of *the algorithm's own past posteriors*. This recursive nature of EPP, which it inherits from the MPP algorithm, makes it hard to analyse.

We denote by $\mathrm{P}^{\mathsf{fr}}_{\mathfrak{H}}$ and $\mathrm{P}^{\mathsf{sl}}_{\mathfrak{H}}$ the probability distributions on $\langle Q_t, E_t, X_t\rangle_{t\in\mathbb{N}}$ defined by EPP-FREEZING and EPP-SLEEPING on EHMM $\mathfrak{H}$ and mixing scheme $\beta$. For both $\mathsf{f/s} \in \{\mathsf{sl}, \mathsf{fr}\}$

$$\mathrm{P}^{\mathsf{f/s}}_{\mathfrak{H}}(q_{1:T}, e_{1:T}, x_{1:T}) \;=\; \prod_{t\in 1:T} \mathrm{p}^{\mathsf{alg}}_t(q_t, e_t, x_t).$$

**Table 1** Mixing schemes

| Mixing scheme | $\beta_{t+1}(t)$ | $\beta_{t+1}(j)$ for $0 \le j < t$ |
|---|---|---|
| Yesterday | 1 | 0 |
| Fixed Share($\alpha$) | $1-\alpha$ | $\alpha$ if $j = 0$ and 0 o.w. |
| Uniform past($\alpha$) | $1-\alpha$ | $\alpha/t$ |
| Decaying past($\alpha, \gamma$) | $1-\alpha$ | $\alpha(t-j)^{-\gamma}/Z_t$ |

### 5.2.1 Representation Invariance

Let $\mathfrak{H}^1$ and $\mathfrak{H}^2$ be EHMMs that are based on the same set of experts $\mathcal{E}$, but have different state spaces. We call $\mathfrak{H}^1$ and $\mathfrak{H}^2$ *equivalent* if $\mathfrak{H}^1(e_{1:T}) = \mathfrak{H}^2(e_{1:T})$ for all $e_{1:T}$. Consequently, equivalent EHMMs assign the same probability $\mathfrak{H}^1(x_{1:T}) = \mathfrak{H}^2(x_{1:T})$ to all data $x_{1:T}$, hence the difference between $\mathfrak{H}^1$ and $\mathfrak{H}^2$ is merely a matter of *representation*. As an important sanity check, we need to verify that EPP on either EHMM issues the same predictions.

**Theorem 2** (Invariance)**.** *Let* f/s *denote either* fr *or* sl*. Fix equivalent EHMMs $\mathfrak{H}^1$ and $\mathfrak{H}^2$. Then for all data $x_{1:T}$*

$$\mathrm{P}^{\mathsf{f/s}}_{\mathfrak{H}^1}(x_{1:T}) \;=\; \mathrm{P}^{\mathsf{f/s}}_{\mathfrak{H}^2}(x_{1:T}).$$

*Proof.* Given in Appendix C. □

Thus, from the perspective of predictive performance, the difference between $\mathfrak{H}^1$ and $\mathfrak{H}^2$ is irrelevant. Of course, it does matter for the computational cost of EPP, see §5.2.3.

### 5.2.2 Mixing Schemes

Bousquet and Warmuth [1] provide an extensive discussion of possible mixing schemes. Their loss bounds for various schemes carry over directly to our setting. It is interesting, however, to analyse the running times of the Fixed-Share to *uniform past* and to *decaying past* mixing schemes for EPP. For further information we refer the reader to [1].

Both schemes (see Table 1) depend on a *switching rate* $\alpha \in [0, 1]$, which determines whether to continue with yesterday's posterior or switch back to an earlier one: $\beta_{t+1}(t) = 1 - \alpha$ and $\sum_{0\le j<t} \beta_{t+1}(j) = \alpha$.

**Uniform Past** Given the choice to switch back, the uniform past mixing scheme gives equal weights to the entire past: $\beta_{t+1}(j) = \alpha/t$ for $0 \le j < t$.

**Decaying Past** Instead, the decaying past scheme assigns larger weight to the recent past: $\beta_{t+1}(j) = \alpha(t-j)^{-\gamma}/Z_t$ for $0 \le j < t$, where $Z_t = \sum_{0\le j<t}(t-j)^{-\gamma}$ is a normalising constant and $\gamma \ge 0$ is a parameter that determines the rate of decay.

### 5.2.3 Running Times

Appendix A provides a detailed comparison of the running times and space requirements of EPP and the forward algorithm. The upshot is that for the uniform past mixing scheme the sleeping variant of EPP is as efficient as the forward algorithm, in terms of both running time and space requirements; the freezing variant is equally efficient if the set of hidden states $\mathcal{Q}$ is finite, but may be a factor $O(T)$ less efficient on $T$ outcomes for countably infinite $\mathcal{Q}$. The decaying past mixing scheme is a factor $O(T)$ less efficient (for both time and space) than uniform past in all cases, but may

be approximated by a scheme described in [1] that reduces this factor to $O(\log T)$.

### 5.3 Loss Bound

This bounds relates the performance of EPP-FREEZING and EPP-SLEEPING (defined in Algorithm 1) to that of $\mathfrak{H}^{\mathsf{fr}}_{\mathbb{C}}$ and $\mathfrak{H}^{\mathsf{sl}}_{\mathbb{C}}$ for all partitions $\mathbb{C}$ *jointly*.

**Theorem 3** (EPP Loss Bounds)**.** *For both* $\mathsf{f/s} \in \{\mathsf{fr}, \mathsf{sl}\}$ *and any mixing scheme* $\beta$*, data* $x_{1:T}$ *and expert predictions* $\mathrm{p}^{\mathcal{E}}_{1:T}$

$$\mathrm{P}^{\mathsf{f/s}}_{\mathfrak{H}}(x_{1:T}) \;\geq\; \sum_{\mathbb{C}} \beta(\mathbb{C}) \mathfrak{H}^{\mathsf{f/s}}_{\mathbb{C}}(x_{1:T}). \tag{5}$$

*Proof.* Given in Appendix B. □

Using this bound, we can relate the predictive performance of EPP-SLEEPING and EPP-FREEZING to that of $\mathfrak{H}^{\mathsf{sl}}_{\mathbb{C}}$ and $\mathfrak{H}^{\mathsf{fr}}_{\mathbb{C}}$ for any reference partition $\mathbb{C}$.

**Corollary 2.** $\mathrm{P}^{\mathsf{f/s}}_{\mathfrak{H}}(x_{1:T}) \;\geq\; \beta(\mathbb{C}) \mathfrak{H}^{\mathsf{f/s}}_{\mathbb{C}}(x_{1:T})$.

From the brutal way in which Corollary 2 was obtained, we may expect to often do much better in practice; *many* partitions may contribute significantly to (5).

### 5.4 Recovering MPP

We now substantiate our claim that EPP generalises MPP by proving that MPP results from running EPP-FREEZING or EPP-SLEEPING on the Bayesian EHMM (Example 4.2).

**Theorem 4.** *Let* $\mathfrak{H}$ *be the Bayesian EHMM with initial distribution* $w$*, and let* $\mathrm{P}^{\mathsf{MPP}}_{w}$ *denote the probability distribution defined by MPP with prior* $w$*. Then for all data* $x_{1:T}$

$$\mathrm{P}^{\mathsf{fr}}_{\mathfrak{H}}(x_{1:T}) \;=\; \mathrm{P}^{\mathsf{sl}}_{\mathfrak{H}}(x_{1:T}) \;=\; \mathrm{P}^{\mathsf{MPP}}_{w}(x_{1:T}).$$

*Proof.* The difference between freezing and sleeping (line 8) evaporates since state evolution is the identity operation. By identifying states and experts the MPP algorithm [1, Figure 1] remains. □

The theorem does not require the set of experts $\mathcal{E}$ to be finite. If $\mathcal{E}$ is infinite (or too large), MPP is intractable. Still, a small EHMM may exist that implements Bayes (say with the uniform prior) on $\mathcal{E}$, and we can use EPP-SLEEPING (which is faster than EPP-FREEZING) for sequential prediction. For example, we may implement MPP on the infinite set of Bernoulli experts (cf the example in the introduction) efficiently, in time $O(T^2)$, using EPP-SLEEPING on the *universal element-wise mixture* EHMM of [9, §4.1].

#### 5.4.1 Improved MPP Loss Bound

[1, Theorem 7] (our Theorem 1) bounds the overhead of MPP over Freund's scheme in terms of $\beta(\mathbb{C})$, the complexity of the reference partition $\mathbb{C}$ according to the mixing scheme $\beta$. A more general bound follows directly from Theorems 3 and 4:

**Corollary 3.** $\mathrm{P}^{\mathsf{MPP}}_{w}(x_{1:T}) \;\geq\; \sum_{\mathbb{C}} \beta(\mathbb{C}) \mathrm{P}^{\mathsf{B}}_{\mathbb{C}}(x_{1:T})$.

Even with a fixed reference partition $\mathbb{C}$ in mind, we get a better bound by considering small modifications of $\mathbb{C}$, e.g. finer partitions or partitions that disagree about a single round.

**Adversarial Experts** For each number of rounds $T$ one can construct a set of $T$ base experts and data $x_{1:T}$ such that the loss of Freund's scheme under the MPP interpretation is infinite for all partitions except the finest one. We simply have expert $t$ suffer infinite loss in all rounds other than $t$. In this pathological case the bounds in Theorem 1 for that partition and Corollary 3 are equal and tight.

#### 5.4.2 Is EPP strictly more general than MPP?

A natural question is whether either EPP-SLEEPING or EPP-FREEZING can be implemented using MPP on a rich set of meta-experts. To preclude the trivial answer that regards either algorithm as a single meta-expert, we ask for a fixed construction that works for all mixing schemes.

**Sleeping** For any EHMM $\mathfrak{H}$, EPP-SLEEPING can be reduced to MPP on meta-experts. Let the set of meta-experts be $\mathcal{Q}^{\infty}$, the set of paths through the hidden states of $\mathfrak{H}$. Each meta-expert $q_{\mathbb{N}}$ predicts $x_t$ using the $\mathrm{p}^{q_t}_{\downarrow}$-mixture of base expert predictions. We set the prior $w$ in MPP equal to the marginal probability measure of $\mathfrak{H}$ on paths (as determined by $\mathrm{p}_{\circ}$ and $\mathrm{p}_{\rightarrow}$). We omit the proof that the predictions made by MPP on these meta-experts with prior $w$ are equal to those made by EPP on $\mathfrak{H}$.

**Freezing** The next example shows that EPP-FREEZING really is more general than MPP. Fix two experts $\mathcal{E} = \{\mathrm{a}, \mathrm{b}\}$. Consider the EHMM $\mathfrak{H}$ that predicts the first outcome using expert a, and the second outcome using expert b, i.e. $\mathcal{Q} = \mathcal{E}$, and $\mathrm{p}_{\circ}(\mathrm{a}) = \mathrm{p}^{\mathrm{a}}_{\rightarrow}(\mathrm{b}) = \mathrm{p}^{q}_{\downarrow}(q) = 1$. Running EPP-FREEZING on $\mathfrak{H}$ results in $\pi_2(\mathrm{b}) = \pi_1(\mathrm{a}) = 1$, so that the first outcome is predicted using expert a, and the second outcome is predicted using the $\beta_2$-mixture of experts. Thus any candidate meta-expert *must* predict the first outcome using base expert a. But that means that for MPP with prior $w$ on meta-experts, the loss update has no effect, so that $w = \pi_1 = \pi_2 = \lambda_2$. Hence the second outcome will be predicted according to the prior mixture of experts. Since $\beta_2$ is arbitrary and $w$ is fixed, there can be no general scheme to reduce EPP-FREEZING to MPP.

## 6 Other Loss Functions

We will now show how the EPP algorithm for logarithmic loss can be directly translated into an algorithm with corresponding loss bound for any other mixable loss function. The same construction works for any logarithmic loss algorithm that predicts according to a mixture of the experts' predictions at each trial and whose predictions only depend on the experts' past losses on outcomes that actually occurred.

**Mixability** A loss function $\ell\colon \mathcal{A} \times \mathcal{X} \to [0, \infty]$ is called *$\eta$-mixable* for $\eta > 0$ if any distribution $p$ on experts $\mathcal{E}$ can be mapped to a single action $\mathrm{Pred}(p) \in \mathcal{A}$ in a way that guarantees that

$$\ell\big(\mathrm{Pred}(p), x\big) \leq -\tfrac{1}{\eta} \log \mathop{\mathrm{E}}_{e \sim p} \Big[ \exp\big(-\eta\, \ell(a^e, x)\big) \Big] \tag{6}$$

for all outcomes $x \in \mathcal{X}$ and expert predictions $a^{\mathcal{E}}$. It is called *mixable* if it is $\eta$-mixable for some $\eta > 0$ [2]. Mixability ensures that expert predictions for $\ell$ loss can be mixed in essentially the same way as for log loss.

For example, logarithmic loss itself is 1-mixable. And for $\mathcal{A} = [0,1]$ and $\mathcal{X} = \{0,1\}$ the *square loss* $\ell(a,x) \coloneqq (a-x)^2$ is 2-mixable and the *Hellinger loss* $\ell(a,x) \coloneqq ((\sqrt{1-x}-\sqrt{1-a})^2+(\sqrt{x}-\sqrt{a}))/2$ is $\sqrt{2}$-mixable.[7, 2]

**The Benefits of Lying** Given data $x_{1:t}$ and expert predictions $a^{\mathcal{E}}_{1:t}$, let $\ell^e_{1:t} \coloneqq \ell(a^e_1,x_1),\ldots,\ell(a^e_t,x_t)$ denote the sequence of losses of expert $e$, and let $\ell\ell^{\mathcal{E}}_{1:t}$ denote these losses jointly for all experts. In the special case that $\ell$ is the logarithmic loss we write $\ell\ell^e_{1:t}$ and $\ell\ell^{\mathcal{E}}_{1:t}$, respectively.

Suppose ALG is an algorithm for log loss that predicts each outcome $x_t$ by mixing the experts' predictions $\mathrm{p}^{\mathcal{E}}_t$ according to the distribution $\mathrm{p}^{\mathsf{alg}}_t[x_{<t}, \ell\ell^{\mathcal{E}}_{<t}]$ on *experts*. The square-bracket expression indicates that $\mathrm{p}^{\mathsf{alg}}_t$ may depend on the past outcomes $x_{1:t-1}$ and the losses of the experts on these outcomes, but not on the experts' past or current predictions in any other way. Following this convention, the algorithm predicts $x_t$ using:

$$\mathrm{p}^{\mathsf{alg}}_t[x_{<t}, \ell\ell^{\mathcal{E}}_{<t}](x_t) \;\coloneqq\; \sum_e \mathrm{p}^{\mathsf{alg}}_t[x_{<t}, \ell\ell^{\mathcal{E}}_{<t}](e)\mathrm{p}^e_t(x_t).$$

Now for any game with $\eta$-mixable loss $\ell$ and the same set of experts $\mathcal{E}$, we can derive from ALG an algorithm $\text{ALG}^\eta_\ell$ that predicts $x_t$ according to

$$a_t^{\mathsf{alg}^\eta_\ell} \;\coloneqq\; \mathrm{Pred}\left(\mathrm{p}^{\mathsf{alg}}_t[x_{<t}, \eta\,\ell^{\mathcal{E}}_{<t}]\right).$$

Note that $\text{ALG}^\eta_\ell$ is lying to ALG: while ALG thinks it is playing a game for log loss in which experts have incurred log losses $\eta\,\ell^{\mathcal{E}}_{<t}$, in reality $\text{ALG}^\eta_\ell$ is playing a game for loss $\ell$ and is feeding ALG fake inputs and redirecting ALG's outputs. Let us now analyse the loss of the derived algorithm $\text{ALG}^\eta_\ell$.

**Lemma 1** (Other Loss Functions)**.** *Suppose* ALG *is an algorithm for logarithmic loss that predicts according to* $\mathrm{p}^{\mathsf{alg}}_t[x_{<t}, \ell\ell^{\mathcal{E}}_{<t}]$ *at each time $t$, $\ell$ is an $\eta$-mixable loss function, and $f(x_{1:T}, \ell\ell^{\mathcal{E}}_{1:T})$ is an arbitrary function that maps outcomes and expert losses to real numbers. Then any log loss bound for* ALG *of the form*

$$-\log \mathrm{P}^{\mathsf{alg}}(x_{1:T}) \;\le\; f(x_{1:T}, \ell\ell^{\mathcal{E}}_{1:T}) \qquad \textit{for all } \mathrm{p}^{\mathcal{E}}_{1:T}, \quad (7)$$

*directly implies the $\ell$ loss bound for* $\text{ALG}^\eta_\ell$*:*

$$\ell(a^{\mathsf{alg}^\eta_\ell}_{1:T}, x_{1:T}) \;\le\; \tfrac{1}{\eta} f(x_{1:T}, \eta\,\ell^{\mathcal{E}}_{1:T}) \quad \textit{for all } a^{\mathcal{E}}_{1:T}. \quad (8)$$

*Proof.* Construct a log loss game in which at any time $t$ each expert $e$ predicts according to a distribution $\mathrm{p}^e_t$ such that $\mathrm{p}^e_t(x_t) = \exp(-\eta\,\ell^e_t)$ for the actual outcome $x_t$ and $\mathrm{p}^e_t$ is arbitrary on other outcomes such that $\sum_{x_t} \mathrm{p}^e_t(x_t) = 1$. In this game the log loss of ALG is

$$-\log \mathrm{P}^{\mathsf{alg}}(x_{1:T}) = \sum_{t\in 1:T} -\log \mathrm{p}^{\mathsf{alg}}_t[x_{<t}, \eta\,\ell^{\mathcal{E}}_{<t}](x_t).$$

By $\eta$-mixability of $\ell$

$$\begin{aligned}\ell(a^{\mathsf{alg}^\eta_\ell}_{1:T}, x_{1:T}) \;&=\; \sum_{t\in 1:T} \ell\left(\mathrm{Pred}\left(\mathrm{p}^{\mathsf{alg}}_t[x_{<t}, \eta\,\ell^{\mathcal{E}}_{<t}]\right), x_t\right)\\ &\le\; \tfrac{1}{\eta}\sum_{t\in 1:T} -\log \mathrm{p}^{\mathsf{alg}}_t[x_{<t}, \eta\,\ell^{\mathcal{E}}_{<t}](x_t). \quad (9)\end{aligned}$$

Combining with (7) and (9) completes the proof. □

Algorithms that satisfy the requirements of the lemma include Bayes, follow the (perturbed) leader, the forward algorithm, MPP and EPP. An algorithm that does not satisfy them is the last-step minimax algorithm [13], because it takes into account the experts' predictions on outcomes that do not occur.

In the literature it is common to construct algorithms for arbitrary mixable losses and point out their probabilistic interpretation for the special case of log loss [7, 8, 1]. Instead, we have proceeded the other way around: first we derived results for log loss and then we showed that they generalise to other losses. This allowed us to draw on concepts and results from probability theory like conditional probabilities, HMMs and the forward algorithm, without reproving them in a more general setting.

Lemma 1 generalises results by Vovk [17], who shows that the most important loss bounds for Bayes with logarithmic loss can actually also be derived for arbitrary mixable losses. Our algorithm ALG plays a role similar to his APA algorithm.

## 7 Discussion

**Relearning vs Continuing to Learn** Corollary 2 bounds the regret of EPP with respect to a reference partition $\mathbb{C}$ by $-\log\beta(\mathbb{C})$. Consider the asymptotic behaviour of this bound if $\mathbb{C}$ has infinitely many shifts. (A shift occurs when $\mathrm{prev}^{\mathbb{C}}(t+1) \neq t$.) For both decaying past with $\gamma \le 1$ (e.g. following recommendations in [1]) and uniform past (see Table 1) $\max_{0\le j<t}\beta_{t+1}(j)$ goes to zero as a function of $t$. Thus, the cost per shift (be it to continue an earlier cell or to start a new one) grows without bound. On the other hand for fixed share $\beta_{t+1}(0) = \alpha$ for all $t$, hence fixed share can start a new cell at fixed cost. It depends on the structured expert whether continuing previously selected cells at increasing cost is advantageous over relearning from scratch after each shift at fixed cost. For EHMM experts with a finite state space $\mathcal{Q}$ (including Bayes), relearning from scratch will cost at most a factor $|\mathcal{Q}|$ over learning on. This factor is constant, so that fixed share will eventually win.

## 8 Conclusion

We revisited Freund's problem, which asks for a strategy for prediction with expert advice that suffers low additional loss compared to Freund's reference scheme. We discussed the solution by Bousquet and Warmuth, which interprets the experts as black boxes. We proposed a new interpretation of Freund's scheme which is natural for learning experts, namely to train experts on the subsequence on which they are evaluated. This allows the reference scheme to exploit local patterns in the data, and thus makes the problem harder.

We solved Freund's problem for structured experts that are represented as EHMMs, building on the work of Bousquet and Warmuth. We showed that our prediction strategies are efficient, and have desirable loss bounds that apply to all mixable losses.

## A Running Times

We compare the running times on $T$ outcomes of EPP and the forward algorithm, with respect to an arbitrary EHMM

$\mathfrak{H}$ with a countable set of hidden states $\mathcal{Q}$. For simplicity we assume that the sets of experts $\mathcal{E}$ and outcomes $\mathcal{X}$ are finite.

Let $Q_t$ denote the hidden state of $\mathfrak{H}$ at time $t$, and let $\mathrm{p}_\circ$, $\mathrm{p}_\rightarrow$ and $\mathrm{p}_\downarrow$ denote $\mathfrak{H}$'s other components. Both algorithms base their predictions on a distribution $\lambda_t$ on $Q_t$ at time $t$, but differ in how they update $\lambda_t$ after observing $x_t$. As the number of computations for this step depends on the size of the support of $\lambda_t$ and on $\mathrm{p}_\rightarrow$, we will need the following concepts. For any probability distribution p on $\mathcal{Q}$, let $\mathrm{Sp}(\mathrm{p}) = \{q \in \mathcal{Q} \mid \mathrm{p}(q) > 0\}$ denote its support. We recursively define $\mathcal{Q}_t$, the set of states reachable in exactly $t$ steps, and $\mathcal{Q}_{\leq t}$, the set of states reachable in at most $t$ steps, by

$$\mathcal{Q}_1 \coloneqq \mathrm{Sp}(\mathrm{p}_\circ), \quad \mathcal{Q}_{t+1} \coloneqq \bigcup_{q \in \mathcal{Q}_t} \mathrm{Sp}(\mathrm{p}_\rightarrow^q), \quad \mathcal{Q}_{\leq t} \coloneqq \bigcup_{i \in 1:t} \mathcal{Q}_i.$$

Obviously, $\mathcal{Q}_t \subseteq \mathcal{Q}_{\leq t} \subseteq \mathcal{Q}$ holds for all $t$. Let $g(S) \coloneqq \sum_{q \in S} |\mathrm{Sp}(\mathrm{p}_\rightarrow^q)|$ be the number of outgoing transitions from any set of states $S \subseteq \mathcal{Q}$.

### A.1 Forward

The forward algorithm computes $\lambda_{t+1}$ by conditioning $\lambda_t$ on $x_t$ and applying the transition function $\mathrm{p}_\rightarrow$. As $\lambda_t$ has support $\mathcal{Q}_t$, the forward algorithm requires $O\big(g(\mathcal{Q}_t)\big)$ work per time step, and $O\big(|\mathcal{Q}_t| + |\mathcal{Q}_{t+1}|\big)$ space. Notice that, for finite $\mathcal{Q}$, the number of transitions is bounded by $g(S) \leq |\mathcal{Q}|^2$ for any $S$. A rough upper bound on the total running time of forward on $T$ outcomes is therefore $O\big(|\mathcal{Q}|^2 T\big)$, which is linear in $T$.

### A.2 EPP

The EPP algorithm comes in two variants: one for sleeping and one for freezing. For sleeping the order of the running time is determined by the evolution of past posteriors (line 8 in Algorithm 1); for freezing, which skips line 8, either computation of $\lambda_t$ (line 3) or of the next posterior (line 7) is the dominant step. The main difference for the running times of the two variants, however, is that in sleeping $\pi_j$ has support $\mathcal{Q}_t$ at any time $t$, whereas for freezing $\pi_j$ has support $\mathcal{Q}_{\leq j}$.

#### A.2.1 Uniform Past

For the uniform past mixing scheme, one can keep track of $\sum_{j=0}^{t} \pi_j(q_t)$ to speed up computation of $\lambda_{t+1}$.

**Sleeping** This even works for sleeping, because applying the state evolution to this sum in line 8 of the algorithm is equivalent to applying it to the individual $\pi_j$ and then summing. Consequently, sleeping requires $O\big(g(\mathcal{Q}_t)\big)$ work and $O\big(|\mathcal{Q}_t| + |\mathcal{Q}_{t+1}|\big)$ space per time step, which makes it as efficient as the forward algorithm.

**Freezing** For freezing, computing the next posterior (line 7) determines the running time. It requires $O\big(g(\mathcal{Q}_{\leq t})\big)$ work and $O\big(|\mathcal{Q}_{\leq t+1}|\big)$ space per time step. Depending on the EHMM $\mathfrak{H}$, this may be significantly slower than the forward algorithm. First, for finite $\mathcal{Q}$, each of $\mathcal{Q}_t$, $\mathcal{Q}_{\leq t}$ and $\mathcal{Q}$ have size $O(1)$ in $t$, and freezing runs in time $O(T)$, just like the forward algorithm. Second, for infinite $\mathcal{Q}$, $\mathcal{Q}_{\leq t}$ may be unbounded as a function of $t$. Still, on $T$ outcomes

$$\sum_{t \in 1:T} g(\mathcal{Q}_{\leq t}) \;\leq\; T g(\mathcal{Q}_{\leq T}) \;\leq\; T \sum_{t \in 1:T} g(\mathcal{Q}_t),$$

which implies that freezing is no more than a factor $T$ slower than the forward algorithm.

#### A.2.2 Decaying Past

For the decaying past scheme the relative mixing weights of any two past posteriors change from $\beta_t$ to $\beta_{t+1}$, which prevents us from summing them as for uniform past. Implementing decaying past therefore slows down both the evolution of past posteriors and computation of $\lambda_t$ by a factor of $O(t)$, and increases the required space by the same factor. Fortunately, however, the decaying past scheme can be approximated using a logarithmic number of uniform blocks, as described in Appendix C of [1]. This reduces the slowdown factor from $O(t)$ to $O(\log t)$.[1] Thus, both for sleeping and for freezing, approximated decaying past is only a factor $O(\log T)$ slower than uniform past on $T$ outcomes, and requires only a factor $O(\log T)$ more space.

## B Loss Bounds

We identify $\lambda_t$ with the EHMM on $\langle Q_i, E_i, X_i \rangle_{i \geq t}$ with initial distribution $\lambda_t$, and with the transition and production functions of $\mathfrak{H}$. So in particular $\lambda_1 = \mathfrak{H}$. For convenience, we shorten $(\lambda_t)^{\mathsf{fr}}_{\mathcal{C}}(x_{\mathcal{C}})$ to $\lambda_t^{\mathsf{fr}}(x_{\mathcal{C}})$ and $(\lambda_t)^{\mathsf{sl}}_{\mathcal{C}}(x_{\mathcal{C}})$ to $\lambda_t^{\mathsf{sl}}(x_{\mathcal{C}})$. Thus, among others, $\lambda_t(x_t) = \lambda_t^{\mathsf{sl}}(x_t) = \lambda_t^{\mathsf{fr}}(x_t)$.

**Lemma 2.** *For any $\mathcal{C} \subseteq t{:}T$, interpreting $\lambda_0(\cdot|x_0)$ as $\lambda_1$,*

$$\lambda_t^{\mathsf{f/s}}(x_{\mathcal{C}}) = \sum_{j \in 0:t-1} \beta_t(j) \lambda_j^{\mathsf{f/s}}(x_{\mathcal{C}}|x_j).$$

*Proof.* Let $\pi_j^t$ denote the past posterior $\pi_j$ at the beginning of round $t$. Thus for freezing $\pi_j^t = \pi_j$, and for sleeping $\pi_j^t$ is $\pi_j$ evolved $t - j$ steps. Then by definition $\lambda_t(x_{\mathcal{C}}) = \sum_{j=0}^{t-1} \beta_t(j) \pi_{j+1}^t(x_{\mathcal{C}})$. The operations $(\cdot)^{\mathsf{fr}}$ and $(\cdot)^{\mathsf{sl}}$ distribute over taking mixtures. The lemma follows from the fact that $(\pi_j^t)^{\mathsf{sl}}(x_{\mathcal{C}}) = \pi_j^{\mathsf{sl}}(x_{\mathcal{C}})$ and $(\pi_j^t)^{\mathsf{fr}}(x_{\mathcal{C}}) = \pi_j^{\mathsf{fr}}(x_{\mathcal{C}})$. □

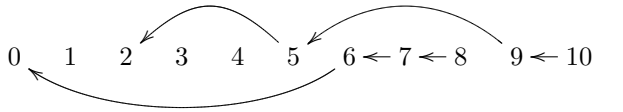
*Proof of Theorem 3.* For any $t$, we view the mixing scheme $\beta_t$ as defining the distribution of a randomized choice $j_t \in 0{:}(t-1)$ for the predecessor of the $t$th outcome. Let $j_{>k} \coloneqq j_{k+1:T} = (j_{k+1}, \ldots, j_T)$ denote a vector of the choices beyond turn $k$. Unfortunately, some choices of $j_{>k}$ are inconsistent with any partition, because an element can only have one successor in a partition. Thus $j_{>k}$ is inconsistent with any partition if $j_m = j_n > 0$ for $k < m \neq n \leq T$. Let the predicate $I(j_{>k})$ be true iff $j_{>k}$ is consistent with some partition.

Some elements of $j_{>k}$ may indicate the start of a new cell of the partition. Let $S(j_{>k})$ denote the set of times when $j_{>k}$ prescribes to start a new cell, i.e. $S(j_{>k}) \coloneqq \{t \in k+1{:}T \mid j_t = 0\}$. For an example, consult Figure 7.

Consistent values of $j_{>k}$ specify the last part of a partition. For any $1 \leq t \leq k$, we may ask which of the times

[1] In [1] it is suggested to weight each block of posteriors $\pi_{[j_1, j_2-1]}$ by $(j_2 - j_1)\beta_t(j_1)$. It seems that a marginal improvement is possible by weighting by $\sum_{j_1 \leq j < j_2} \beta_t(j)$ instead, which can be implemented equally efficiently for decaying past.

**Figure 7** Notation example. $T = 10$, $k = 4$, $j_{>k} = (2, 0, 6, 7, 5, 9)$, $S(j_{>k}) = \{6\}$, $R_2(j_{>k}) = \{2, 5, 9, 10\}$.




$k+1{:}T$ will be put in the same cell as $t$. Let $R_t(j_{>k})$ denote this set, including $t$. For convenience, we abbreviate

$$\beta(j_{>k}) \coloneqq \prod_{t \in k+1:T} \beta_t(j_t),$$

$$W(j_{>k}) \coloneqq \prod_{i \in S(j_{>k})} \lambda_1^{\mathsf{f/s}}(x_{R_i(j_{>k})}), \qquad \text{and}$$

$$U_l(j_{>k}) \coloneqq \prod_{i \in 1:l} \lambda_i^{\mathsf{f/s}}(x_{R_i(j_{>k})}) \qquad \text{for all } l \leq k,$$

to name the intermediate debris arising from the incremental reduction of $\mathrm{P}_{\mathfrak{H}}^{\mathsf{f/s}}(x_{1:T})$. $W$-terms deal with cells that are completely specified by $j_{>k}$, while $U$-terms keep track of the remaining partially specified cells. The proof proceeds by downward induction on $k$, with induction hypothesis

$$\prod_{i \in 1:T} \lambda_i(x_i) \geq \sum_{j_{>k} : I(j_{>k})} \beta(j_{>k}) W(j_{>k}) U_k(j_{>k}). \quad (10)$$

For the base case $k = T$ the hypothesis holds with equality, and for $k = 0$ the hypothesis is equivalent to the desired result (5). It remains to verify that it holds for $k - 1 \geq 0$ if it holds for $k$. To this end, fix $k \geq 1$. To prove (10), it suffices to show that for consistent $j_{>k}$

$$W(j_{>k}) U_k(j_{>k}) \geq \sum_{j_k : I(j_{\geq k})} \beta_k(j_k) W(j_{\geq k}) U_{k-1}(j_{\geq k}),$$

where $j_{\geq k}$ denotes $j_{k:T}$, i.e. $j_k$ followed by $j_{>k}$. We expand the last factor of $U_k(j_{>k})$ using Lemma 2, and bound

$$\begin{aligned} U_k(j_{>k}) &= \sum_{j_k \in 0:k-1} \beta_k(j_k) \lambda_{j_k}^{\mathsf{f/s}}(x_{R_k(j_{>k})} | x_{j_k}) U_{k-1}(j_{>k}) \\ &\geq \sum_{j_k : I(j_{\geq k})} \beta_k(j_k) \lambda_{j_k}^{\mathsf{f/s}}(x_{R_k(j_{>k})} | x_{j_k}) U_{k-1}(j_{>k}). \end{aligned}$$

Observe that $R_t(j_{>k}) = R_t(j_{\geq k})$ for all $1 \leq t < k$ except $t = j_k$. There are two cases. If $j_k = 0$, then $U_{k-1}(j_{>k}) = U_{k-1}(j_{\geq k})$ and $W(j_{>k}) \lambda_1^{\mathsf{f/s}}(x_{R_k(j_{>k})}) = W(j_{\geq k})$ ; on the other hand if $j_k > 0$ then $W(j_{>k}) = W(j_{\geq k})$. For consistent $j_{\geq k}$, $U_{k-1}(j_{>k})$ contains the factor $\lambda_{j_k}^{\mathsf{f/s}}(x_{j_k})$, which implies that

$$\lambda_{j_k}^{\mathsf{f/s}}(x_{R_k(j_{>k})} | x_{j_k}) U_{k-1}(j_{>k}) = U_{k-1}(j_{\geq k}). \qquad \square$$

# C Invariance

*Proof of Theorem 2.* Let $\mu^1$ and $\mu^2$ be distributions on $\mathcal{Q}^1$ and $\mathcal{Q}^2$. We overload notation, and write $\mu^1$ and $\mu^2$ for the EHMMs $\mathfrak{H}^1$ and $\mathfrak{H}^2$ with initial distribution replaced by $\mu^1$ and $\mu^2$. Recall that $\mu^1$ and $\mu^2$ are equivalent if $\mu^1(e_{1:T}) = \mu^2(e_{1:T})$ for all $e_{1:T}$. Thus, $\mathfrak{H}^1$ and $\mathfrak{H}^2$ are equivalent iff $\mathrm{p}_\circ^1$ and $\mathrm{p}_\circ^2$ are equivalent.

To prove the theorem, we need to prove that equivalence is preserved by all the operations that EPP performs, i.e. taking mixtures, performing loss update and performing state evolution. Mixtures of equivalent distributions are equivalent, since mixing and marginalisation commute. For loss update, note that $\mathrm{p}_1^{e_1}(x_1) = \mu^1(x_1 | e_{1:T}) = \mu^2(x_1 | e_{1:T})$ for all $\mathrm{p}_1^{\mathcal{E}}$ and all $e_{1:T}$. Finally, for state evolution, the claim follows from $(\mathrm{p}_\rightarrow \circ \mu)(e_{1:T}) = \mu(E_{2:T+1} = e_{1:T})$. $\square$

# References


[1] O. Bousquet and M. K. Warmuth. Tracking a small set of experts by mixing past posteriors. *Journal of Machine Learning Research*, 3:363–396, 2002.

[2] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[3] Z. Ghahramani and G. E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12(4):831–864, 2000.

[4] Z. Ghahramani and M. I. Jordan. Factorial hidden markov models. *Machine Learning*, 29(2-3):245–273, 1997.

[5] R. B. Gramacy, M. K. Warmuth, S. A. Brandt, and I. Ari. Adaptive caching by refetching. In *In Advances in Neural Information Processing Systems 15*, pages 1465–1472. MIT Press, 2002.

[6] J. Hannan. Approximation to Bayes risk in repeated play. In *Contributions to the Theory of Games*, volume 3, pages 97–139. Princeton University Press, 1957.

[7] D. Haussler, J. Kivinen, and M. K. Warmuth. Sequential prediction of individual sequences under general loss functions. *IEEE Transactions on Information Theory*, 44(5):1906–1925, 1998.

[8] M. Herbster and M. K. Warmuth. Tracking the best expert. *Machine Learning*, 32:151–178, 1998.

[9] W. M. Koolen and S. de Rooij. Combining expert advice efficiently. In *Proceeding of the 21st Annual Conference on Learning Theory*, pages 275–286, 2008.

[10] N. Landwehr. Modeling interleaved hidden processes. In *Proceedings of the 25th international conference on Machine learning*, pages 520–527, New York, NY, USA, 2008. ACM.

[11] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[12] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, issue 2, pages 257–285, 1989.

[13] E. Takimoto and M. Warmuth. The last-step minimax algorithm. In *Proceedings of the 13th Annual Conference on Computational Learning Theory*, pages 100–106, 2000.

[14] T. van Erven, P. D. Grünwald, and S. de Rooij. Catching up faster in Bayesian model selection and model averaging. In *Advances in Neural Information Processing Systems 20*, 2008.

[15] P. A. Volf and F. M. Willems. Switching between two universal source coding algorithms. In *Proceedings of the Data Compression Conference, Snowbird, Utah*, pages 491–500, 1998.

[16] V. Vovk. Aggregating strategies. In *Proceedings of the third Annual Conference on Computational Learning Theory*, pages 371–383, 1990.

[17] V. Vovk. Derandomizing stochastic prediction strategies. *Machine Learning*, 35:247–282, 1999.