Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, 1997.

Using and combining predictors that specialize

Yoav Freund Robert E. Schapire Yoram Singer
AT&T Labs*
600 Mountain Avenue, Murray Hill, NJ 07974
{yoav, schapire, singer}@research.att.com

Manfred K. Warmuth
University of California
Santa Cruz, CA 95064
manfred@cse.ucsc.edu

Abstract. We study online learning algorithms that predict by combining the predictions of several subordinate prediction algorithms, sometimes called "experts." These simple algorithms belong to the multiplicative weights family of algorithms. The performance of these algorithms degrades only logarithmically with the number of experts, making them particularly useful in applications where the number of experts is very large. However, in applications such as text categorization, it is often natural for some of the experts to abstain from making predictions on some of the instances. We show how to transform algorithms that assume that all experts are always awake to algorithms that do not require this assumption. We also show how to derive corresponding loss bounds. Our method is very general, and can be applied to a large family of online learning algorithms. We also give applications to various prediction models including decision graphs and "switching" experts.

1 Introduction

We study online learning algorithms that predict by combining the predictions of several subordinate prediction algorithms, sometimes called "experts." Starting with the work of Vovk [19] and Littlestone and Warmuth [14], many algorithms have been developed in recent years which use multiplicative weight updates. These algorithms enjoy theoretical performance guarantees which can be proved without making any statistical assumptions. Such results can be made meaningful in a non-statistical setting by proving that the performance of the master algorithm can never be much worse than that of the best expert. Furthermore, the dependence of such a bound on the number of experts is only logarithmic, making such algorithms applicable even when the number of experts is enormous.

In this paper, we study an extension of the online prediction framework first proposed by Blum [1]. The added feature is that we allow experts to abstain from making a prediction. Experts that are given the possibility to abstain are called *specialists*, because we think of them as making their prediction only when the instance to be predicted falls within their area of expertise. We say that a specialist is *awake* when it makes a prediction and that it is *asleep* otherwise. We refer to the conventional framework as the *insomniac* framework since it is a special case in which all specialists are awake all the time.

An important real-world application of prediction for which specialists are very useful is in the field of information retrieval. Consider the problem of predicting the category to which a news article belongs (such as "politics," "weather," "sports," etc.) based on the appearance of words in the given article. We can think of each word as a feature and represent each article as a vector of features. In this case the number of features is huge (on the order of 10^5), which makes multiplicative weight-update algorithms very attractive. It is intuitively clear that most of the features are relevant only to the small subset of the documents in which they appear. A natural way for using this intuition is to use specialists that predict when a specific word or combination of words appear in the document and are asleep otherwise. This leads to very efficient algorithms that can deal with huge vocabularies and make very good predictions. This was demonstrated by Cohen and Singer [3] who used one of the specialist algorithms described in this paper for such a text-classification task. Thus, our results generalize a theoretical foundation to an algorithm that has already been shown to be of practical value.

In the first part of this paper, we give a general transformation for converting an insomniac algorithm into the specialist framework and how the corresponding bounds can also be transformed. This transformation can be applied to a large family of learning problems and algorithms, including all those that fall within Vovk's [18] very general framework of online learning, as well as the algorithms belonging to the "exponentiated gradient" family of algorithms introduced by Kivinen and Warmuth [12]. The feature common to the analysis of all these algorithms is that they use an amortized analysis in which relative entropy is the potential function.

In the second part of the paper we show that using specialists is a powerful way for *decomposing* complex prediction

*AT&T Labs is planning to move from Murray Hill in 1997. The new address will be: 180 Park Avenue, Florham Park, NJ 07932-0971.

problems. The naive solution to these prediction problems uses a very large set of experts, making the calculations of the prediction computationally infeasible. We show how a large set of experts can be represented using a much smaller set of specialists. Each expert corresponds to a subset of the specialists which take turns in making their predictions. Only a small fraction of the specialists are involved in producing each prediction, which reduces the computational load even further.

Specifically, we apply this decomposition to the problem of predicting almost as well as the best pruning of a decision graph. This generalizes previous work on predicting almost as well as the best pruning of a decision tree [21, 10].

We also apply our methods to the problem of predicting in a model in which the "best" expert may change with time. We derive a specialist-based algorithm for this problem that is as fast as the best known algorithm of Herbster and Warmuth [11] and achieves almost as good a loss bound. However, unlike their algorithm, ours does not require prior knowledge of the length of the sequence and the number of switches.

2 The specialist framework

We now give a formal definition of the framework. We define online learning with specialists as a game that is played between the prediction algorithm and an adversary. We assume that there are N specialists, indexed by $\{1, \ldots, N\}$. We assume that predictions and outcomes are real-valued numbers from a bounded range $[0, 1]$.[1] We define a *loss function* $\mathrm{L} : [0,1] \times [0,1] \to [0, \infty)$ that associates a non-negative loss to each pair of prediction and outcome.

The game proceeds in iterations $t = 1, \ldots, T$, each consisting of the following five steps:

1. The adversary chooses a set $E_t \subseteq \{1, \ldots, N\}$ of specialists that are awake at iteration t.
2. The adversary chooses a prediction $x_{t,i}$ for each awake specialist $i \in E_t$.
3. The algorithm chooses its own prediction $\hat{y}_t$.
4. The adversary chooses an outcome y_t.
5. The algorithm suffers loss $\ell_A^t = \mathrm{L}(\hat{y}_t, y_t)$ and each of the awake specialists suffers loss $\ell_i^t = \mathrm{L}(x_{t,i}, y_t)$. Specialists that are asleep suffer no loss.

The performance of an algorithm is measured in terms of its total loss $L_A = \sum_{t=1}^{T} \ell_A^t$. We are interested in bounds that hold for *any adversarial strategy*. As the adversary chooses the outcome after the algorithm made its prediction, it can clearly inflict on the algorithm a large loss on each iteration. In order to give a meaningful bound, we consider the *difference* between the total loss of the algorithm and the total loss of the experts. The total loss of insomniac algorithms is usually compared to the loss of the best expert. Such a comparison does not make sense in the specialists framework because it is possible that no expert is awake all of the time. Instead, we compare the total loss of our algorithm to the loss of the best fixed *mixture* of the experts, as defined precisely below. Our goal is to derive bounds which guarantee that the performance of the algorithm will be good whenever there exists some mixture of the specialists that is good. Thus, the adversary cannot make the algorithm suffer large loss unless it inflicts large loss on all mixtures of specialists.[2]

This mixture of experts can be done in two ways, and we consider both in this paper. We denote by Δ_N the set of probability vectors of dimension N.

• **Comparison to average loss:** In the easier type of comparison, we compare the total loss of the algorithm to

$$\min_{\mathbf{u} \in \Delta_N} \sum_{t=1}^{T} \mathrm{L}_{\mathbf{u}}^{I}(\mathbf{x}_t, y_t) \tag{1}$$

where

$$\mathrm{L}_{\mathbf{u}}^{I}(\mathbf{x}_t, y_t) \doteq \frac{\sum_{i \in E_t} u_i \, \mathrm{L}(x_{t,i}, y_t)}{\sum_{i \in E_t} u_i} \, .$$

The expression $\sum_{t=1}^{T} \mathrm{L}_{\mathbf{u}}^{I}(\mathbf{x}_t, y_t)$ describes the total loss of an algorithm that at each iteration t predicts by randomly *choosing one* of the specialists in E_t according to the fixed distribution $\mathbf{u}$ restricted to E_t and re-normalized. Equation (1) defines the total loss of the best distribution $\mathbf{u}$, which suffers the minimal loss for the particular sequence. As this optimal $\mathbf{u}$ is not known in advance, it is impossible to actually achieve a total loss of (1), and all we can hope for is to guarantee that the loss of our algorithms is never much larger than it.

• **Comparison to average prediction:** In this case we compare the total loss of the algorithm to

$$\min_{\mathbf{u} \in \Delta_N} \sum_{t=1}^{T} \mathrm{L}_{\mathbf{u}}^{II}(\mathbf{x}_t, y_t) \tag{2}$$

where

$$\mathrm{L}_{\mathbf{u}}^{II}(\mathbf{x}_t, y_t) \doteq \mathrm{L}\left(\frac{\sum_{i \in E_t} u_i x_{t,i}}{\sum_{i \in E_t} u_i}, y_t \right)$$

This has a similar interpretation to the average loss comparison but in this case we consider the loss of an idealized algorithm which predicts with the combined prediction of the awake specialists, rather than choosing one of them at

[1] In two of the three cases we present here, the outcomes must lie in $\{0, 1\}$. While some results can be presented in the much more general online prediction framework of Vovk [18], we chose to simplify this paper by making these more restrictive choices.

[2] This definition is similar in motivation to the definition of regret in the statistical analysis of prediction algorithms. However, unlike in that case, no statistical assumptions are made here regarding the mechanism that is generating the sequence. The bounds here hold for any sequence of outcomes.

random and predicting with its prediction. Since in most interesting cases the loss function is convex, bounds of this form imply bounds of the previous form but not vice versa. Bounds of this second form are harder to achieve.

In his work on predicting using specialists [1], Blum proves a bound on the performance of a variant of the Winnow algorithm [13]. This algorithm is used for making binary predictions and Blum made the additional assumption that a non-empty subset of the specialists never make a mistake. It is assumed that at any iteration at least one of these infallible specialists is awake. This is a special case of our framework in which there exists a vector $\mathbf{u}$ (which has non-zero components on the infallible subset of specialists) such that the loss associated with this vector is zero.[3]

3 Design and analysis of specialist algorithms

In this section we show how to transform insomniac learning algorithms into the specialist framework. We start with a simple case and then describe a general transformation which we then apply to other, more complex cases.

A few preliminaries: Recall that Δ_N denotes the set of probability vectors of dimension N, i.e., $\Delta_N = \{\mathbf{p} \in [0,1]^N : \sum_i p_i = 1\}$. For two probability vectors $\mathbf{u}, \mathbf{v} \in \Delta_N$, the relative entropy, written $\mathbf{RE}\left(\mathbf{u} \| \mathbf{v}\right)$ is $\sum_i u_i \ln(u_i/v_i)$. (We follow the usual convention that $0 \ln 0 = 0$.) For probability vector $\mathbf{u} \in \Delta_N$ and a set $E \subseteq \{1, \ldots, N\}$, we define $u(E) = \sum_{i \in E} u_i$.

3.1 Log loss

One of the simplest and best known online prediction algorithms is the Bayes algorithm, which has been rediscovered many times, for instance, in the context of universal coding, Bayesian estimation and investment management [4, 6, 7]. In this case, the predictions are from the range $[0,1]$, the outcomes are from $\{0,1\}$ and the loss is the log loss, or coding length, defined as

$$\mathrm{L}(\hat{y}, y) = \begin{cases} -\ln \hat{y} & \text{if } y = 1 \\ -\ln(1-\hat{y}) & \text{if } y = 0. \end{cases}$$

Note that this loss is always nonnegative, but may be infinite (for instance, if $\hat{y} = 0$ and $y = 1$).

$\mathbf{Bayes}$ algorithm is described on the left side of Figure 1. This algorithm maintains a probability vector $\mathbf{p}_t$ over the N experts.[4] On each round t, each expert i provides a prediction $x_{t,i} \in [0,1]$. The Bayes algorithm combines these by taking their average with respect to $\mathbf{p}_t$ and predicting $\hat{y}_t = \mathbf{p}_t \cdot \mathbf{x}_t$. The outcome y_t then defines the loss of each expert and of the master algorithm. The weights are then updated so as to increase the weights of the experts with relatively small loss, thereby ensuring that their predictions will count more on the next round.

In our context the justification for using this algorithm is the following bound on the total loss relative to the loss of the best expert:

$$\begin{aligned} \sum_{t=1}^{T} \mathrm{L}(\hat{y}_t, y_t) &\le \min_{\mathbf{u} \in \Delta_N} \left(\sum_{t=1}^{T} \sum_{i=1}^{N} u_i \mathrm{L}(x_{t,i}, y_t) + \mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_1\right) \right) \\ &\le \min_i \sum_{t=1}^{T} \mathrm{L}(x_{t,i}, y_t) + \ln N \end{aligned} \tag{3}$$

where the last inequality holds if the initial "prior" distribution $\mathbf{p}_1$ is chosen to be uniform. Note that this bound holds for all sequences of expert predictions and outcomes. Also, note that the additional loss grows only logarithmically with the number of experts.

On the right side of Figure 1, we present the algorithm $\mathbf{SBayes}$, an adaptation of the insomniac algorithm $\mathbf{Bayes}$ to the case of specialists. The adaptation is simple: on each round, we treat E_t, the set of specialists that are awake, as if it were the complete set of specialists and leave the weights of the other specialists untouched. We then re-normalize the weights of the awake specialists so that their total weight remains unchanged. The main theorem regarding the performance of $\mathbf{SBayes}$ is a direct adaptation of the well-known theorem regarding $\mathbf{Bayes}$.

Theorem 1 *For any sequence of awake specialists, specialist predictions and outcomes and for any distribution* $\mathbf{u}$ *over* $\{1, \ldots, N\}$*, the loss of* **SBayes** *satisfies*

$$\sum_{t=1}^{T} u(E_t) L(\hat{y}_t, y_t) \le \sum_{t=1}^{T} \sum_{i \in E_t} u_i L(x_{t,i}, y_t) + \mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_1\right).$$

Proof: We show first that

$$\begin{aligned} &\mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_t\right) - \mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_{t+1}\right) \\ &\quad = \quad u(E_t) \mathrm{L}(\hat{y}_t, y_t) - \sum_{i \in E_t} u_i \mathrm{L}(x_{t,i}, y_t). \end{aligned} \tag{4}$$

Consider the change in the relative entropy between $\mathbf{u}$ and $\mathbf{p}_t$ on two consecutive trials:

$$\begin{aligned} \mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_t\right) - \mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_{t+1}\right) &= \sum_{i=1}^{N} u_i \ln \frac{p_{t+1,i}}{p_{t,i}} \\ &= \sum_{i \in E_t} u_i \ln \frac{p_{t+1,i}}{p_{t,i}}. \end{aligned}$$

If $y_t = 1$, then the latter quantity is equal to

$$\begin{aligned} \sum_{i \in E_t} u_i \ln \frac{x_{t,i}}{\hat{y}_t} &= \sum_{i \in E_t} u_i \ln x_{t,i} - u(E_t) \ln \hat{y}_t \\ &= -\sum_{i \in E_t} u_i \mathrm{L}(x_{t,i}, y_t) + u(E_t) \mathrm{L}(\hat{y}_t, y_t). \end{aligned}$$

[3] However, the bounds derived by Blum are not comparable with the bounds given here because he considers bounds which have no dependency on the total number of specialists, while all our bounds have some dependence on this number.

[4] This distribution over experts is often called the posterior distribution, and it has a natural probabilistic interpretation. However, as in this work we make no probabilistic assumptions, the posterior distribution should be regarded simply as real-valued weights that are used by the prediction algorithm.

Parameters: Prior distribution $\mathbf{p}_1 \in \Delta_N$; number of trials T.

Algorithm Bayes
Do for $t = 1, 2, \ldots, T$

1. Predict with the weighted average of the experts predictions:
$$\hat{y}_t = \sum_{i=1}^{N} p_{t,i} x_{t,i}$$
2. Observe outcome y_t
3. Calculate a new posterior distribution:
$$p_{t+1,i} = \begin{cases} \frac{p_{t,i} x_{t,i}}{\hat{y}_t} & \text{if } y_t = 1 \\ \frac{p_{t,i}(1 - x_{t,i})}{1 - \hat{y}_t} & \text{if } y_t = 0. \end{cases}$$

Algorithm SBayes
Do for $t = 1, 2, \ldots, T$

1. Predict with the weighted average of the predictions of the awake specialists:
$$\hat{y}_t = \frac{\sum_{i \in E_t} p_{t,i} x_{t,i}}{\sum_{i \in E_t} p_{t,i}}$$
2. Observe outcome y_t
3. Calculate a new posterior distribution:
$$\text{If } i \in E_t, \text{ then } p_{t+1,i} = \begin{cases} \frac{p_{t,i} x_{t,i}}{\hat{y}_t} & \text{if } y_t = 1 \\ \frac{p_{t,i}(1 - x_{t,i})}{1 - \hat{y}_t} & \text{if } y_t = 0. \end{cases}$$
Otherwise, $p_{t+1,i} = p_{t,i}$.

Figure 1: The Bayes algorithm and the Bayes algorithm for specialists.

The proof is similar when $y_t = 0$. We thus get Equation (4).

Summing this equality for $t = 1, \ldots, T$ and using the fact that the relative entropy is always positive we get

$$\begin{aligned} \mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_1\right) &\geq \mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_1\right) - \mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_{T+1}\right) \\ &= \sum_{t=1}^{T} u(E_t) \mathrm{L}(\hat{y}_t, y_t) - \sum_{t=1}^{T} \sum_{i \in E_t} u_i \mathrm{L}(x_{t,i}, y_t). \end{aligned}$$

Rearranging terms then gives the statement of the theorem. ■

If, in addition to the conditions of Theorem 1, $u(E_t) = U$ for all $1 \leq t \leq T$ then we get the following bound that is easier to interpret than the theorem:

$$\sum_{t=1}^{T} \mathrm{L}(\hat{y}_t, y_t) \leq \sum_{t=1}^{T} \frac{\sum_{i \in E_t} u_i \mathrm{L}(x_{t,i}, y_t)}{\sum_{i \in E_t} u_i} + \frac{\mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_1\right)}{U}. \tag{5}$$

The first term on the right hand side of this inequality is equal to the expected loss of a prediction algorithm that predicts according to the distribution vector $\mathbf{u}$ as follows. On iteration t the algorithm chooses one of the awake experts according to the distribution defined by restricting $\mathbf{u}$ to the set E_t. It then predicts with the prediction of the chosen expert. Equation (5) shows that the total loss of our algorithm is never much larger than the total loss incurred by using *any* such fixed $\mathbf{u}$. It also shows that the gap is proportional to the distance between the prior distribution $\mathbf{p}_1$ and the comparison distribution $\mathbf{u}$ and is inversely proportional to the fraction of the specialists that are awake at each iteration.

Lastly, note that algorithm $\mathbf{Bayes}$ is a special case of $\mathbf{SBayes}$ where $E_t = \{1, \ldots, N\}$ for all t. Thus, the bound given in Equation (3) is derived from Equation (5) by setting $U = 1$.

3.2 The general case

In this section, we generalize the method suggested in the last section and show how it can be applied to a large family of on-line algorithms. We give a general method for converting an insomniac on-line algorithm in this family, along with its relative loss bound, into the corresponding specialist algorithm and loss bound.

We focus in this section on algorithms which, like $\mathbf{Bayes}$, maintain a distribution vector $\mathbf{p}_t \in \Delta_N$. In general, such algorithms consist of two parts:

1. a prediction function $\mathrm{pred}_N : \Delta_N \times [0,1]^N \to [0,1]$ which maps the current weight vector $\mathbf{p}_t$ and instance $\mathbf{x}_t$ to a prediction $\hat{y}_t$; and
2. an update function $\mathrm{update}_N : \Delta_N \times [0,1]^N \times [0,1] \to \Delta_N$ which maps the current weight vector $\mathbf{p}_t$, instance $\mathbf{x}_t$ and outcome y_t to a new weight vector $\mathbf{p}_{t+1}$.

When clear from context, we drop the subscript on pred_N and update_N.

The functioning of such an algorithm is shown on the left side of Figure 2.

The conversion of such an algorithm to the specialist framework in which some of the experts may be sleeping is fairly straightforward. First, for any nonempty subset $E \subseteq \{1, \ldots, N\}$ of awake specialists and instance $\mathbf{x} \in [0,1]^N$, let $\mathbf{x}^E \in [0,1]^{|E|}$ denote the restriction of $\mathbf{x}$ to the components of E. Formally, if $E = \{i_1, \ldots, i_{|E|}\}$ with $i_1 < \cdots < i_{|E|}$ then $x^E_j = x_{i_j}$. Similarly, let $\mathbf{p}^E \in \Delta_{|E|}$ denote the restriction of $\mathbf{p}$ to E but now the components are also normalized. Thus, $p^E_j = p_{i_j} / \sum_{i \in E} p_i$.

The specialist version of our abstract on-line learning algorithm is shown on the right side of Figure 2. The prediction depends only on the awake specialists, and is given by

Insomniac algorithm
Do for $t = 1, 2, \ldots, T$

1. Observe $\mathbf{x}_t$.
2. Predict $\hat{y}_t = \text{pred}(\mathbf{p}_t, \mathbf{x}_t)$.
3. Observe outcome y_t and suffer loss $\mathrm{L}(\hat{y}_t, y_t)$.
4. Calculate the new weight vector $\mathbf{p}_{t+1} = \text{update}(\mathbf{p}_t, \mathbf{x}_t, y_t)$

Specialist algorithm
Do for $t = 1, 2, \ldots, T$

1. Observe E_t and $\mathbf{x}_t^{E_t}$.
2. Predict $\hat{y}_t = \text{pred}(\mathbf{p}_t^{E_t}, \mathbf{x}_t^{E_t})$.
3. Observe outcome y_t and suffer loss $\mathrm{L}(\hat{y}_t, y_t)$.
4. Calculate the new weight vector $\mathbf{p}_{t+1}$ so that it satisfies the following:
   (a) $p_{t+1,i} = p_{t,i}$ for $i \notin E_t$
   (b) $\mathbf{p}_{t+1}^{E_t} = \text{update}(\mathbf{p}_t^{E_t}, \mathbf{x}_t^{E_t}, y_t)$
   (c) $\sum_{i=1}^{N} p_{t+1,i} = 1$.

Figure 2: Abstract insomniac and specialist on-line learning algorithms.

$\text{pred}(\mathbf{p}_t^{E_t}, \mathbf{x}_t^{E_t})$. The update rule says to leave the weights of sleeping specialists unchanged, and to modify the weights of awake experts in the natural way. That is, we modify these weights so that $\mathbf{p}_{t+1}^{E_t} = \text{update}(\mathbf{p}_t^{E_t}, \mathbf{x}_t^{E_t}, y_t)$ while meeting the requirement that $\sum_i p_{t+1,i} = 1$ (or equivalently, that $\sum_{i \in E_t} p_{t+1,i} = \sum_{i \in E_t} p_{t,i}$).

It can be verified that, when this transformation is applied to $\mathbf{Bayes}$, the resulting algorithm is exactly $\mathbf{SBayes}$.

Analysis

As in the case of $\mathbf{Bayes}$, a large family of on-line learning algorithms can be analyzed by examining $\mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_t\right)$, the relative entropy between a comparison distribution vector $\mathbf{u}$ and the algorithm's weight vector $\mathbf{p}_t$. For instance, the key fact in the analysis of $\mathbf{Bayes}$ is the following:

$$\mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_t\right) - \mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_{t+1}\right) = \mathrm{L}(\hat{y}_t, y_t) - \sum_{i=1}^{N} u_i \mathrm{L}(x_{t,i}, y_t).$$

This is a trivial special case of Equation (4) with all specialists awake.

The analysis of many other insomniac algorithms is based on a similar core inequality of the form

$$\mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_t\right) - \mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_{t+1}\right) \geq a\mathrm{L}(\hat{y}_t, y_t) - b\mathrm{L}_{\mathbf{u}}(\mathbf{x}_t, y_t). \tag{6}$$

Here, a and b are positive constants which depend on the specific on-line learning problem, $\mathrm{L}(\hat{y}, y)$ is the loss of the algorithm, and $\mathrm{L}_{\mathbf{u}}(\mathbf{x}, y)$ is the comparison loss of vector $\mathbf{u}$, which in this paper will always be either $\mathrm{L}_{\mathbf{u}}^{I}(\mathbf{x}, y)$ or $\mathrm{L}_{\mathbf{u}}^{II}(\mathbf{x}, y)$ as defined in the introduction. For instance, for $\mathbf{Bayes}$, $a = b = 1$, L is log loss, and our bound is with respect to $\mathrm{L}_{\mathbf{u}}^{I}$.

Equation (6) immediately gives a bound on the cumulative loss of the algorithm since, by summing over $t = 1, \ldots, T$ we get

$$\begin{aligned} \mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_1\right) &\geq \mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_1\right) - \mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_{T+1}\right) \\ &\geq a \sum_{t=1}^{T} \mathrm{L}(\hat{y}_t, y_t) - b \sum_{t=1}^{T} \mathrm{L}_{\mathbf{u}}(\mathbf{x}_t, y_t) \end{aligned}$$

so

$$\sum_{t=1}^{T} \mathrm{L}(\hat{y}_t, y_t) \leq \frac{b}{a} \sum_{t=1}^{T} \mathrm{L}_{\mathbf{u}}(\mathbf{x}_t, y_t) + \frac{1}{a}\mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_1\right). \tag{7}$$

Suppose now that we move to the specialist algorithm. We have that

$$\begin{aligned} &\mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_t\right) - \mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_{t+1}\right) = \\ &= \sum_i u_i \ln \frac{p_{t+1,i}}{p_{t,i}} = \sum_{i \in E_t} u_i \ln \frac{p_{t+1,i}}{p_{t,i}} \\ &= u(E_t) \left(\mathbf{RE}\left(\mathbf{u}^{E_t} \| \mathbf{p}_t^{E_t}\right) - \mathbf{RE}\left(\mathbf{u}^{E_t} \| \mathbf{p}_{t+1}^{E_t}\right) \right). \end{aligned}$$

Assuming Equation (6) holds, this last term is at least

$$u(E_t) \left(a\mathrm{L}(\hat{y}_t, y_t) - b\mathrm{L}_{\mathbf{u}^{E_t}}(\mathbf{x}_t^{E_t}, y_t) \right)$$

by construction of $\hat{y}_t$ and $\mathbf{p}_{t+1}$. Thus, we have proved the following general bound which is the main result of this section:

$$\sum_{t=1}^{T} u(E_t)\mathrm{L}(\hat{y}_t, y_t) \leq \tag{8}$$
$$\tfrac{b}{a} \textstyle\sum_{t=1}^{T} u(E_t)\mathrm{L}_{\mathbf{u}^{E_t}}(\mathbf{x}_t^{E_t}, y_t) + \tfrac{1}{a}\mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_1\right) \quad .$$

In short, we have shown that essentially any online insomniac algorithm with a bound of the form given in Equation (7) has a corresponding specialist algorithm with a bound of the form given in Equation (8), provided that the insomniac bound was proved using the inequality in Equation (6).

We now give several applications of this bound for specific loss functions. In addition to those included in this abstract, the method can be applied to many other online algorithms, including all the algorithms derived for the expert setting [19, 8, 2]. This is possible because the analysis of all of these algorithms can be rewritten using the relative entropy as a measure of progress.

Parameters: Prior distribution $\mathbf{p}_1 \in \Delta_N$;
learning rate $\eta > 0$; number of trials T.

Algorithm SAbs
Do for $t = 1, 2, \ldots, T$

1. Predict with:

$$\hat{y}_t = F_\eta \left(\frac{\sum_{i \in E_t} p_{t,i} x_{t,i}}{\sum_{i \in E_t} p_{t,i}} \right)$$

where $F_\eta : [0,1] \to [0,1]$ is any function which satisfies, for all $0 \le r \le 1$:

$$1 + \frac{\ln((1-r)e^{-\eta} + r)}{2 \ln \frac{2}{1+e^{-\eta}}} \le F_\eta(r) \le \frac{-\ln(1 - r + re^{-\eta})}{2 \ln \frac{2}{1+e^{-\eta}}}$$

2. Observe outcome y_t and incur loss $\mathrm{L}(\hat{y}_t, y_t) = |\hat{y}_t - y_t|$.

3. Calculate a new posterior distribution: if $i \in E_t$

$$p_{t+1,i} = p_{t,i} e^{-\eta |x_{t,i} - y_t|} \frac{\sum_{j \in E_t} p_{t,j}}{\sum_{j \in E_t} p_{t,j} e^{-\eta |x_{t,j} - y_t|}}$$

Otherwise, $p_{t+1,i} = p_{t,i}$.

Figure 3: The multiplicative weights algorithm for specialists and absolute loss.

3.3 Absolute loss

The absolute loss function is defined by $\mathrm{L}(\hat{y}, y) = |\hat{y} - y|$, where, in this section, we assume that $y \in \{0, 1\}$. For this loss function, it is natural to interpret $\hat{y} \in [0, 1]$ as a randomized prediction in $\{0, 1\}$ which is 1 with probability $\hat{y}$ and 0 otherwise. Then the loss $|\hat{y} - y|$ is the probability of a mistake, and the cumulative loss measures the expected number of mistakes in a sequence of randomized predictions.

For the absolute loss, we can apply the transformation of Section 3.2 to the algorithm of Cesa-Bianchi et al. [2] which is based on the work of Vovk [19]. This yields an algorithm that is similar but somewhat more complex than $\mathbf{SBayes}$, which we call $\mathbf{SAbs}$, and which is shown in Figure 3. Like $\mathbf{SBayes}$, $\mathbf{SAbs}$ maintains a weight for each specialist which it updates by multiplicative factors after each iteration. There are two main differences between $\mathbf{SAbs}$ and $\mathbf{SBayes}$. First, $\mathbf{SAbs}$ has a parameter $\eta > 0$, sometimes called a "learning rate," that has to be set before the sequence is observed (see Cesa-Bianchi et al. [2] for a detailed discussion of how to choose η). Second, the prediction is not a weighted average of the predictions of the experts, but rather a function of this average which also depends on η.

To analyze $\mathbf{SAbs}$, we first rewrite the analysis of this algorithm [19, 2] using the notation from Section 3.2. The coefficients in the instantiation of Equation (6) that apply to this case depend on η and are

$$a_\eta = 2 \ln \frac{2}{1 + e^{-\eta}} \quad \text{and} \quad b_\eta = \eta \,. \tag{9}$$

It is easy to verify that in this case the two types of comparison losses are equal: $\sum_i u_i |x_i - y| = |\mathbf{u} \cdot \mathbf{x} - y|$.

Applying the general reduction from Section 3.2 to this case we get the following bound:

$$\sum_{t=1}^{T} u(E_t) |\hat{y}_t - y_t| \le \tag{10}$$
$$\frac{1}{a_\eta} \left(\eta \sum_{t=1}^{T} u(E_t) |\mathbf{u} \cdot \mathbf{x}_t - y_t| + \mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_1\right) \right) .$$

Parameters: Prior distribution $\mathbf{p}_1 \in \Delta_N$;
learning rate $\eta > 0$; number of trials T.

Algorithm SEG
Do for $t = 1, 2, \ldots, T$

1. Predict with:

$$\hat{y}_t = \frac{\sum_{i \in E_t} p_{t,i} x_{t,i}}{\sum_{i \in E_t} p_{t,i}}$$

2. Observe outcome y_t and incur loss $\mathrm{L}(\hat{y}_t, y_t) = |\hat{y}_t - y_t|$.

3. Calculate a new posterior distribution: if $i \in E_t$

$$p_{t+1,i} = p_{t,i} e^{-2\eta x_{t,i} (\hat{y}_t - y_t)} \frac{\sum_{j \in E_t} p_{t,j}}{\sum_{j \in E_t} p_{t,j} e^{-2\eta x_{t,i} (\hat{y}_t - y_t)}}$$

Otherwise, $p_{t+1,i} = p_{t,i}$.

Figure 4: The exponentiated gradient algorithm for specialists and square loss.

3.4 Square loss

We next consider the square loss $\mathrm{L}(\hat{y}, y) = (\hat{y} - y)^2$. Using the algorithm for on-line prediction with square loss described by Vovk [19], we can derive an algorithm whose bound is in terms of the comparison loss $\mathrm{L}^I_{\mathbf{u}}(\mathbf{x}, y)$. In this section, we show how to get a more powerful bound in terms of $\mathrm{L}^{II}_{\mathbf{u}}(\mathbf{x}, y)$ using a different family of algorithms, called the *exponentiated gradient* ($\mathbf{EG}$) algorithms. This family was introduced by Kivinen and Warmuth [12] and is derived and analyzed using the relative entropy. It thus fits within the framework of Section 3.2.

The $\mathbf{EG}$ algorithm is similar to the algorithms based on Vovk's work in that they maintain one weight per input and update these weights multiplicatively. The main difference is that instead of having the loss in the exponent of the update factor, we have the gradient of the loss.

Applying the transformation of Section 3.2 to $\mathbf{EG}$, we obtain the algorithm $\mathbf{SEG}$ shown in Figure 4. Like $\mathbf{SAbs}$, this algorithm has a parameter $\eta > 0$ that needs to be tuned.

At the core of the relative loss bound for $\mathbf{EG}$, there is again an inequality of the form given in Equation (6). Kivinen and Warmuth [12, Lemma 5.8] prove that such an inequality holds for $a_\eta = \eta$, $b_\eta = \frac{2\eta}{2-\eta}$ and $\mathrm{L}_{\mathbf{u}}^{II}(\mathbf{x}, y) = (\mathbf{u} \cdot \mathbf{x} - y)^2$. We therefore can apply our general results to obtain the bound

$$\sum_{t=1}^{T} u(E_t)(\hat{y}_t - y_t)^2 \leq \frac{2}{2-\eta} \sum_{t=1}^{T} u(E_t)(\mathbf{u}^{E_t} \cdot \mathbf{x}_t^{E_t} - y_t)^2 + \frac{1}{\eta}\mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_1\right)$$

on the relative loss of $\mathbf{SEG}$ with respect to any comparison vector $\mathbf{u}$.

This conversion also works for all other on-line algorithms derivable from the relative entropy such as the versions of $\mathbf{EG}$ where the loss of the algorithm is compared to the loss of the best sigmoided linear neuron [9].

4 Applications

In this section, we describe several applications of the specialist framework. For concreteness, we focus for each application on a specific loss function. The applications described can easily be extended and used with other loss functions.

4.1 Markov models

As an illustration of the specialist methodology, we begin with an application to a simple prediction problem. Suppose we are predicting a binary sequence one bit at a time, and we want to minimize the expected number of mistakes, i.e., the absolute loss. One common approach is to predict according to a k-th order Markov model for some fixed $k > 0$. In this case the prediction of each bit is a function of the k preceding bits. More formally, we want our prediction algorithm to predict almost as well as the best table-lookup function $\mu : \{0,1\}^k \to \{0,1\}$, where we interpret $\mu(s)$ as the prediction of such a function or expert given that s is the preceding sequence of k bits.

Without applying the specialist framework, we could use, for instance, Vovk's [19] (insomniac) expert-prediction algorithm in which we maintain one expert for each table-lookup function. Naively, this would require maintenance of 2^{2^k} weights, all of which must be updated on every trial.

Alternatively, we propose maintaining 2^{k+1} specialists, one for every pair $\langle s, b\rangle$ in $\{0,1\}^k \times \{0,1\}$. Such a specialist is awake if and only if the sequence s exactly matches the preceding k bits, and, when awake, it always predicts b.

This set up requires maintenance of only 2^{k+1} weights. Furthermore, since only two specialists are awake on each round, the time to formulate each prediction and to update the weights is $O(1)$ per round.

To analyze this algorithm, we wish to compare the absolute loss of our algorithm to the absolute loss of the "best" table-lookup function μ. To do so, let the comparison vector $\mathbf{u}$ be uniform over the set of 2^k specialists identified by μ, i.e., the set $\{\langle s, b\rangle : \mu(s) = b\}$. Clearly, on each round, exactly one of these is awake so $u(E_t) = 2^{-k}$ for all t. Also, note that the prediction associated with $\mathbf{u}$ is identical to that of μ. If we choose $\mathbf{p}_1$ to be uniform over all of the defined specialists, then $\mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_1\right) = \ln 2$. Equation (10) then implies immediately that the loss of our algorithm is at most $(\eta/a_\eta)\mathrm{L}^* + (1/a_\eta)2^k \ln 2$ where L^* is the loss of the best table-lookup function, and a_η and b_η are as defined in Equation (10). This bound coincides exactly with the bound which would be obtained using the more naive approach of maintaining an expert for each of the 2^{2^k} table-lookup functions.

We included this example as a simple illustration of the general method. The result is not new; for instance, the same loss bound and time and space complexity can be achieved using a variant of Cover and Shenhar's [5] method of partitioning the data sequence. However, as will be seen in the next sections, we can apply the specialist framework to much more powerful models and derive algorithms that are, to our knowledge, more efficient than the best existing algorithms based on the experts approach.

4.2 Decision graphs

The Markov models described in the previous section are a special case of decision trees which are a special case of decision graphs. In this section we describe decision graphs and prunings of decision graphs. We give an efficient prediction algorithm, based on specialists, which predicts almost as well as the best pruning of a decision graph.

In this section, we use the log loss. On each iteration t, the prediction algorithm receives an instance z^t. We will be interested in predictions computed by decision graphs. A *decision graph* $\mathcal{G}$ is a directed acyclic graph with interior nodes and terminal nodes, and a designated start node. The interior nodes are associated with tests on the input instance z^t. Each interior node has two outgoing edges, one for each possible outcome of the test. Each terminal node is associated with a prediction in $[0,1]$. The prediction associated with an instance is calculated in the natural way. Starting from the start node, the graph is traversed by performing the test associated with the current node, selecting the edge that corresponds to the outcome of the test, and moving to the node pointed to by the selected edge. This process continues until a terminal node is reached. The prediction associated with this terminal node is the prediction of the graph.

For instance, for the graph on the left in Figure 5, given the instance $z = 010$, the terminal node reached is node D whose prediction is 0.7.

When decision graphs are very large, it is sometime advantageous to stop the decision process before reaching a terminal node and instead associate the prediction with an in-

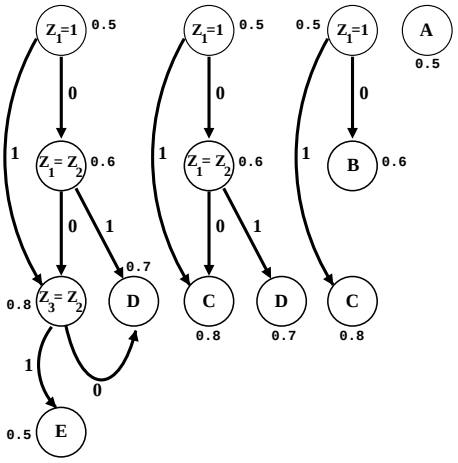

Figure 5: A decision graph and its possible prunings.

ternal node. We call the decision graph that is derived in such a way a *pruning* of the original decision graph. An important example is the well-studied [16, 20, 15, 17, 21] variable-length Markov model, in which the order of decisions is fixed in advance, but the depth of the decision process might depend on the instance. In other words, as in Section 4.1, decisions are based on the preceding sequence of bits, but the number of bits that are examined may not be the same for all instances.

More precisely, assume now that predictions are associated with *all* of the nodes of $\mathcal{G}$ (including interior nodes). In the context of this paper, a pruning $\mathcal{P}$ of this decision graph is any decision graph that can be generated in the following way. First, a set of pruning nodes is selected. Second, all *edges* that are reachable from the pruning nodes are removed. Finally, all nodes that cannot be reached from the start node are removed and all nodes without outgoing edges are defined to be terminal. Note that the terminal nodes include the pruning nodes but can also include other nodes. Figure 5 shows all of the prunings of the left-most graph.

Our goal is to predict almost as well as the pruning that gives the predictions with the minimum loss on the observed sequence. The naive approach would be to maintain one insomniac expert for each possible pruning and adjust the weight of each pruning based on its performance. However, the number of prunings of a decision graph can be exponentially large in the size of the graph, making this approach computationally infeasible.

Instead, we use the specialist framework by associating a specialist with each edge in the full decision graph. The prediction of a specialist is the prediction of the node pointed to by its corresponding edge. A specialist is awake at time step t if and only the sequence of tests performed on z^t traverses its assigned edge.[5] Clearly, the total number of specialists allocated is equal to the number of edges of the decision graph $\mathcal{G}$, and the time needed to formulate a prediction is the length of the path from the start node to a terminal node.

To analyze the algorithm, we compare the log loss of the algorithm to the loss of any pruning. Let $\mathcal{P}$ be the pruning which achieves the smallest total loss. We say that an edge is a *terminal edge* if it is an ingoing edge of a terminal node. We let the comparison vector $\mathbf{u}$ be uniform over all the terminal edges in $\mathcal{P}$. Let k be the number of terminal edges in $\mathcal{P}$, and let m be the total number of edges in the full decision graph $\mathcal{G}$. On each round, exactly one terminal edge of $\mathcal{P}$ is traversed in $\mathcal{G}$; this follows from the manner in which prunings have been defined. Hence, exactly one specialist in the support set of $\mathbf{u}$ is awake so $u(E_t) = 1/k$ for all t. By construction, the loss of $\mathbf{u}$ is equal to the loss of the predictions computed by pruning $\mathcal{P}$. From Theorem 1, we therefore get that the additional loss of the algorithm relative to the loss of $\mathcal{P}$ is at most $k \cdot \mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_1\right)$. If we choose $\mathbf{p}_1$ to be uniform over all the edges in the full decision graph $\mathcal{G}$ then $\mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_1\right) = \ln(m/k)$, giving an additional loss bound of $k \ln(m/k)$. This bound is essentially optimal for general decision graphs.

In the special case that the decision graph is actually a decision tree, we could instead apply the techniques of Willems, Shtarkov and Tjalkens [21] and Helmbold and Schapire [10]. Their methods also lead to an algorithm for predicting almost as well as the best pruning of the decision tree, but results in an additional loss bound of only $O(k)$ where, as above, k is the number of terminal edges of $\mathcal{P}$, which, for trees, is simply equal to the number of leaves. For trees, our bounds can be improved to $O(k^2)$ which is still inferior to the above bound. However, our method is more general and can be applied not only to decision trees but to any directed acyclic graph.

4.3 Switching experts

In the conventional (insomniac) on-line learning model, we compare the loss of the master algorithm to the loss of the best of N experts for the entire sequence. However, it is natural to expect that different experts will be best for different segments of the data sequence. In this section, we study such a model of prediction in which the "best" expert may change with time.

Specifically, we imagine that the sequence of T prediction rounds is subdivided (in a manner unknown to the learner) into at most k segments where each segment is associated with a unique expert which suffers the minimal loss on the segment. The sequence of segments and its associated sequence of best experts is called a *segmentation*. Our goal is to perform well relative to the best segmentation.

This prediction problem was first studied by Littlestone and Warmuth [14]. Currently, the best of the known algorithms for this problem are due to Herbster and Warmuth [11]. Although the algorithms we obtain are just as efficient as theirs ($O(N)$ time per iteration), our bounds are slightly weaker than Herbster and Warmuth's. However, their algo-

[5] In order to handle degenerate situations, we also assign a specialist to a "dummy" edge that comes in to the start node; this specialist is always awake.

rithms require estimates of k and T and their bounds degrade as the quality of the estimates degrades. Our algorithm does *not* require prior knowledge of k and T.

We use the log loss in this section. We call the N original experts the *ground experts*, and we define a set of higher-level experts called *segmentation experts*. A k-segmentation expert is defined by a segmentation of the sequence into k segments, each associated with a ground expert. That is, each k-segmentation expert is defined by a sequence of switch points $0 = t_0 < t_1 < \cdots < t_k = T$ and a sequence of ground experts $e_1, \ldots, e_k$. Here, the interpretation is that the segmentation expert predicts the same as expert e_i on trials $t_{i-1}+1$ through t_i (inclusive). Our goal is to predict almost as well as the best segmentation expert.

If the algorithm were provided in advance with the number of segments k and the length of the sequence T, then we could keep one weight for each of the exponentially many k-segmentation experts and apply the $\mathbf{Bayes}$ algorithm of Section 3.1. In this case, the additional loss, relative to the best k-segmentation expert, is upper bounded by $k \ln N + (k-1)\ln(T/k)$. Note that this bound coincides with the description length (in nats) of a segmentation expert (when k and T are known), a bound which seems impossible to beat. Herbster and Warmuth's bound is essentially larger than this bound by k, provided that k and T are known ahead of time by their algorithm. Our bound is $k(\ln T + o(\ln T))$ larger than either bound, but our algorithm requires no prior knowledge of T and k.

We now describe our construction of specialists for the switching experts problem. We construct one specialist $S(t_1, t_2, i)$ for each ground expert i and for each pair of positive integers $t_1 \le t_2$. Such a specialist uses the predictions of expert i on rounds t_1 through t_2 (inclusive) and is asleep the rest of the time. We choose the initial weight of this specialist to be $p_1(S(t_1, t_2, i)) = \nu(t_2)/(t_2 N)$ where $\nu(t)$ is any distribution on the natural numbers. It is not hard to show that p_1 sums to one when summed over all of the defined specialists.

With this construction of specialists, we are ready to apply $\mathbf{SBayes}$.[6] Let us first analyze the additional loss of the algorithm. For any k-segmentation expert of the form described above, we can set the comparison vector $\mathbf{u}$ to be uniform over the k specialists naturally associated with the segmentation, namely, $S(1, t_1, e_1), S(t_1+1, t_2, e_2), \ldots, S(t_{k-1}+1, T, e_k)$. Since exactly one of these is awake at each time step, $u(E_t) = 1/k$. Furthermore, note that the prediction associated with $\mathbf{u}$ is identical to that of the k-segmentation expert from which it was derived. Therefore, from Theorem 1, we get that the additional loss incurred by our algorithm relative to any k-segmentation expert is at most

$$\frac{\mathbf{RE}\left(\mathbf{u} \| \mathbf{p}_1\right)}{u(E_t)} = k \ln\left(\frac{N}{k}\right) + \sum_{j=1}^{k} \ln t_j - \sum_{j=1}^{k} \ln \nu(t_j).$$

This bound clearly depends on the choice of ν. For instance, if we choose $\nu(t) = c/(t\,[\ln(t+1)]^2)$ for the appropriate normalizing constant c, then the bound is at most $k \ln(N/k) + 2k \ln T + k \cdot o(\ln T)$.

It is not immediately obvious how to implement this algorithm since it requires maintenance of an infinite number of specialists. We describe below an efficient scheme that requires maintenance of only $O(N)$ weights, and in which predictions and updates also require only $O(N)$ time per round. The main idea is to show that the predictions of $\mathbf{SBayes}$ can be written in the form

$$\hat{y}_t = \frac{\sum_{i=1}^{N} Q_i^t x_{t,i}}{\sum_{i=1}^{N} Q_i^t}$$

where $x_{t,i}$ is the prediction of ground expert i at time t and Q_i^t is the total weight of all specialists associated with ground expert i that are active at time t. We then show how to update these weights efficiently, resulting in the algorithm shown in Figure 6.

Let $p^t_{t_1,t_2,i}$ be the weight maintained by $\mathbf{SBayes}$ for specialist $S(t_1, t_2, i)$ at time t. Then the prediction of our algo-

Parameters: Prior distribution ν over $\mathbb{N}$;
number of trials T.

Specialists Algorithm for Switching Experts
Initialize: $Q_i^t = 1/N$; $F_1 = \sum_{t=1}^{\infty} \nu(t)/(tN)$.
Do for $t = 1, 2, \ldots, T$

1. Predict with the weighted average of the experts predictions:
$$\hat{y}^t = \frac{\sum_{i=1}^{N} Q_i^t x_{t,i}}{\sum_{i=1}^{N} Q_i^t}.$$
2. Observe outcome y^t and incur loss.
3. Calculate new weights:
   (a) $F_{t+1} = F_t - \nu(t)/(tN)$
   (b) $R_i^t = \begin{cases} \frac{x_{t,i}}{\hat{y}_t} & \text{if } y_t = 1 \\ \frac{1-x_{t,i}}{1-\hat{y}_t} & \text{if } y_t = 0. \end{cases}$
   (c) $Q_i^{t+1} = F_{t+1}\left(1 + \frac{R_i^t Q_i^t}{F_t}\right)$.

Figure 6: The SBayes algorithm for switching experts.

[6] Although presented for a finite number of specialists, $\mathbf{SBayes}$ (or any of the other algorithms in this paper) can easily be modified to handle a countably infinite number of specialists.

rithm at time t is

$$\hat{y}_t = \frac{\sum_{i=1}^{N} \sum_{t_1=1}^{t} \sum_{t_2=t}^{\infty} p^t_{t_1,t_2,i} x_{t,i}}{\sum_{i=1}^{N} \sum_{t_1=1}^{t} \sum_{t_2=t}^{\infty} p^t_{t_1,t_2,i}} ,$$

where $x_{t,i}$ is the prediction of ground expert i at time t. Let Q_i^t be the total weight of all specialists associated with ground expert i that are active at time t, that is,

$$Q_i^t = \sum_{t_1=1}^{t} \sum_{t_2=t}^{\infty} p^t_{t_1,t_2,i} .$$

Then,

$$\hat{y}_t = \frac{\sum_{i=1}^{N} Q_i^t x_{t,i}}{\sum_{i=1}^{N} Q_i^t} .$$

Our implementation maintains only the N weights Q_i^t. We now show how to update these weights efficiently.

Let

$$R_i^t = \begin{cases} \frac{x_{t,i}}{\hat{y}_t} & \text{if } y_t = 1 \\ \frac{1-x_{t,i}}{1-\hat{y}_t} & \text{if } y_t = 0 \end{cases} .$$

Then, from the manner in which weights are updated by $\mathbf{SBayes}$, we have that

$$p^t_{t_1,t_2,i} = \frac{\nu(t_2)}{t_2 N} \prod_{s=t_1}^{t-1} R_i^s .$$

Therefore,

$$\begin{aligned} Q_i^t &= \sum_{t_1=1}^{t} \sum_{t_2=t}^{\infty} \frac{\nu(t_2)}{t_2 N} \prod_{s=t_1}^{t-1} R_i^s \\ &= F_t \sum_{t_1=1}^{t} \prod_{s=t_1}^{t-1} R_i^s , \end{aligned}$$

where $F_t = \sum_{t_2=t}^{\infty} \frac{\nu(t_2)}{t_2 N}$. We can thus update Q_i^t using the recurrence

$$\begin{aligned} Q_i^{t+1} &= F_{t+1} \sum_{t_1=1}^{t+1} \prod_{s=t_1}^{t} R_i^s \\ &= F_{t+1} \left(1 + R_i^t \sum_{t_1=1}^{t} \prod_{s=t_1}^{t-1} R_i^s \right) = F_{t+1} \left(1 + \frac{R_i^t Q_i^t}{F_t} \right) . \end{aligned}$$

This update takes $O(1)$ time per weight, assuming that F_1 has been precomputed. The resulting algorithm is shown in Figure 6.

Acknowledgments

Thanks to William Cohen for helpful comments on this work.

References

[1] Avrim Blum. Empirical support for winnow and weighted-majority based algorithms: results on a calendar scheduling domain. In *ml95*, pages 64–72, 1995.

[2] Nicolò Cesa-Bianchi, Yoav Freund, David P. Helmbold, David Haussler, Robert E. Schapire, and Manfred K. Warmuth. How to use expert advice. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 382–391, 1993. To appear, *Journal of the Association for Computing Machinery*.

[3] William W. Cohen and Yoram Singer. Context-sensitive learning methods for text categorization. In *sigir96*, pages 307–315, 1996.

[4] Thomas M. Cover. Universal portfolios. *Mathematical Finance*, 1(1):1–29, January 1991.

[5] Thomas M. Cover and Aaron Shenhar. Compound Bayes predictors for sequences with apparent Markov structure. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-7(6):421–424, June 1977.

[6] Alfredo DeSantis, George Markowsky, and Mark N. Wegman. Learning probabilistic prediction functions. In *Proceedings of the 1988 Workshop on Computational Learning Theory*, pages 312–328, 1988.

[7] Robert G. Gallager. *Information Theory and Reliable Communication*. John Wiley & Sons, 1968.

[8] David Haussler, Jyrki Kivinen, and Manfred K. Warmuth. Tight worst-case loss bounds for predicting with expert advice. In *Computational Learning Theory: Second European Conference, EuroCOLT '95*, pages 69–83. Springer-Verlag, 1995.

[9] David P. Helmbold, Jyrki Kivinen, and Manfred K. Warmuth. Worst-case loss bounds for sigmoided neurons. In *Advances in Neural Information Processing Systems 7*, pages 309–315, 1995.

[10] David P. Helmbold and Robert E. Schapire. Predicting nearly as well as the best pruning of a decision tree. In *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, pages 61–68, 1995.

[11] Mark Herbster and Manfred Warmuth. Tracking the best expert. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 286–294, 1995.

[12] Jyrki Kivinen and Manfred K. Warmuth. Additive versus exponentiated gradient updates for linear prediction. In *stoc95*, pages 209–218, 1995. See also technical report UCSC-CRL-94-16, University of California, Santa Cruz, Computer Research Laboratory.

[13] Nick Littlestone. Learning when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[14] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, 1994.

[15] Neri Merhav, Meir Feder, and Michael Gutman. Some properties of sequential predictors for binary Markov sources. *IEEE Transactions on Information Theory*, 39(3):887–892, May 1993.

[16] Jorma Rissanen and Glen G. Langdon, Jr. Universal modeling and coding. *IEEE Transactions on Information Theory*, IT-27(1):12–23, January 1981.

[17] Dana Ron, Yoram Singer, and Naftali Tishby. Learning probabilistic automata with variable memory length. In *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, pages 35–46, 1994.

[18] V. G. Vovk. A game of prediction with expert advice. In *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, 1995.

[19] Volodimir G. Vovk. Aggregating strategies. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 371–383, 1990.

[20] Marcelo J. Weinberger, Abraham Lempel, and Jacob Ziv. A sequential algorithm for the universal coding of finite-memory sources. *IEEE Transactions on Information Theory*, 38(3):1002–1014, May 1992.

[21] Frans M. J. Willems, Yuri M. Shtarkov, and Tjalling J. Tjalkens. The context tree weighting method: basic properties. *IEEE Transactions on Information Theory*, 41(3):653–664, 1995.