Machine Learning, 32, 151–178 (1998)
© 1998 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Tracking the Best Expert *

MARK HERBSTER AND MANFRED K. WARMUTH (mark|manfred)@cs.ucsc.edu
Department of Computer Science, University of California at Santa Cruz, Applied Sciences Building, Santa Cruz, CA 95064

Editors: Gerhard Widmer and Miroslav Kubat

Abstract. We generalize the recent relative loss bounds for on-line algorithms where the additional loss of the algorithm on the whole sequence of examples over the loss of the best expert is bounded. The generalization allows the sequence to be partitioned into segments, and the goal is to bound the additional loss of the algorithm over the sum of the losses of the best experts for each segment. This is to model situations in which the examples change and different experts are best for certain segments of the sequence of examples. In the single segment case, the additional loss is proportional to $\log n$, where n is the number of experts and the constant of proportionality depends on the loss function. Our algorithms do not produce the best partition; however the loss bound shows that our predictions are close to those of the best partition.

When the number of segments is $k+1$ and the sequence is of length ℓ, we can bound the additional loss of our algorithm over the best partition by $O(k \log n + k \log(\ell/k))$. For the case when the loss per trial is bounded by one, we obtain an algorithm whose additional loss over the loss of the best partition is independent of the length of the sequence. The additional loss becomes $O(k \log n + k \log(L/k))$, where L is the loss of the best partition with $k+1$ segments.

Our algorithms for tracking the predictions of the best expert are simple adaptations of Vovk's original algorithm for the single best expert case. As in the original algorithms, we keep one weight per expert, and spend $O(1)$ time per weight in each trial.

Keywords: on-line learning, amortized analysis, multiplicative updates, shifting, experts

1. Introduction

Consider the following on-line learning model. The learning occurs in a series of trials labeled $1, 2, \ldots, \ell$. In each trial t the goal is to predict the *outcome* $y_t \in [0,1]$ which is received at the end of the trial. At the beginning of trial t, the algorithm receives an n-tuple $\mathbf{x}_t$. The element $x_{t,i} \in [0,1]$ of the n-tuple $\mathbf{x}_t$ represents the prediction of an *expert* $\mathcal{E}_i$ of the value of the outcome y_t on trial t. The algorithm then produces a prediction $\hat{y}_t$, based on the current expert prediction tuple $\mathbf{x}_t$, and on past predictions and outcomes. At the end of the trial, the algorithm receives the outcome y_t. The algorithm then incurs a *loss* measuring the discrepancy between the prediction $\hat{y}_t$ and the outcome y_t. Similarly, each expert incurs a loss as well. A possible goal is to minimize the total loss of the algorithm over all ℓ trials on an arbitrary sequence of instance outcome pairs (such pairs are called examples). Since we make no assumptions about the relationship between the prediction of experts ($\mathbf{x}_t$) and the outcome (y_t), there is always some sequence of y_t that is "far away" from the predictions $\hat{y}_t$ of any particular algorithm. Thus, minimizing the total loss over an arbitrary sequence of examples is an unreasonable goal. A refined relativized goal is to minimize the additional loss of the algorithm over the loss of the best expert on the whole sequence. If all experts have large loss then this goal might actually be easy to achieve,

* An extended abstract appeared in (Herbster & Warmuth, 1995)

since for all algorithms the additional loss over the loss of the best expert may then be small. However, if at least one expert predicts well, then the algorithm must "learn" this quickly and produce predictions which are "close" to the predictions of the best expert in the sense that the additional loss of the algorithm over the loss of the best expert is bounded.

This expert framework might be used in various settings. For example, the experts might predict the chance of rain or the likelihood that the stock market will rise or fall. Another setting is that the experts might themselves be various sub-algorithms for recognizing particular patterns. The "master" algorithm that combines the experts' predictions does not need to know the particular problem domain. It simply keeps one weight per expert, representing the belief in the expert's prediction, and then decreases the weight as a function of the loss of the expert.

Previous work of Vovk (1998) and others (Littlestone & Warmuth, 1994; Haussler, Kivinen & Warmuth, 1998) has produced an algorithm for which there is an upper bound on the additional loss of the algorithm over the loss of the best expert. Algorithms that compare against the loss of the best expert are called STATIC-EXPERT algorithms in this paper. The additional loss bounds for these algorithms have the form $c \ln n$ for a large class of loss functions, where c is a constant which only depends on the loss function L, and n is the number of experts. This class of loss functions contains essentially all common loss functions except for the absolute loss and the discrete loss[1] (counting prediction mistakes), which are treated as special cases (Littlestone & Warmuth, 1994; Vovk, 1995; Cesa-Bianchi, Freund, Haussler, Helmbold, Schapire & Warmuth, 1997). For example, if the loss function is the square or relative entropy loss, then $c = \frac{1}{2}$ or $c = 1$, respectively (see Section 2 for definitions of the loss functions).

In the paper we consider a modification of the above goal introduced by Littlestone and Warmuth (1994), in which the sequence of examples is subdivided into $k+1$ segments of arbitrary length and distribution. Each segment has an associated expert. The sequence of segments and its associated sequence of experts is called a *partition*. The loss of a partition is the sum of the total losses of the experts associated with each segment. The best partition of size k is the partition with $k+1$ segments that has the smallest loss. The modified goal is to perform well relative to the best partition of size k. This goal is to model real life situations where the "nature" of the examples might change and a different expert produces better predictions. For example, the patterns might change and different sub-algorithms may predict better for different segments of the on-line sequence of patterns. We seek to design master algorithms that "track" the performance of the best sequence of experts in the sense that they incur small additional loss over the best partition of size k. If the whole sequence of examples was given ahead of time, then one could compute the best partition of a certain size and the associated experts using dynamic programming. Our algorithms get the examples on-line and never produce the best partition. Even so, we are able to bound the additional loss over the best off-line partition for an arbitrary sequence of examples.

When there are ℓ trials, $k+1$ segments, and n experts, there are $\binom{\ell-1}{k} n(n-1)^k$ distinct partitions. We can immediately get a good bound for this problem by expanding the set of n experts into $\binom{\ell-1}{k} n(n-1)^k = O((n^{k+1}(\frac{e\ell}{k})^k)$ "partition-experts." Each partition-expert represents a single partition of the trial sequence, and predicts on each trial as the expert *associated* with the segment which contains the current trial. Thus, using the STATIC-EXPERT algorithm we obtain a bound of $c \ln \binom{\ell-1}{k} n(n-1)^k \leq c[(k+1)\log n +$

$k \log \frac{\ell}{k} + k]$ of the additional loss of the algorithm over the loss of the best partition. There are two problems: first, the algorithm is inefficient, since the number of partition-experts is exponential in the number of partitions; and second, the bound on the additional loss grows with the sequence length.

We were able to overcome both problems. Instead of keeping one weight for the exponentially many partitions, we can get away with keeping only one weight per expert, as done in the STATIC-EXPERT algorithm. So the "tracking" of the predictions of the best partition is essentially for free. If there are n sub-algorithms or experts whose predictions we want to combine, then as in the STATIC-EXPERT algorithm the new master algorithm takes only $O(n)$ additional time per trial over the time required for simulating the n sub-algorithms.

We develop two main algorithms: the FIXED-SHARE Algorithm, and the VARIABLE-SHARE Algorithm. Both of these are based on the STATIC-EXPERT algorithms which maintain a weight of the form $e^{-\eta T_i}$ for each expert (cf. Littlestone & Warmuth, 1994; Vovk, 1995), where T_i is total past loss of the expert i in past trials. In each trial the master algorithm combines the experts' predictions using the current weights of the experts. When the outcome of the trial is received, we multiply the weight of every expert i by $e^{-\eta L_i}$, where L_i is the loss of expert i in the current trial. We call this update of the weights the *Loss Update*.

We modify the STATIC-EXPERT Algorithm by adding an additional update to obtain our algorithms. Since in our model the best expert may shift over a series of trials, we cannot simply use weights of the form $e^{-\eta T_i}$, because before an expert is optimal for a segment its loss in prior segments may be arbitrarily large, and thus its weight may become arbitrarily small. So we need to modify the STATIC-EXPERT Algorithm so that small weights can be recovered quickly.

For this reason, each expert "shares" a portion of its weight with the other experts after the Loss Update; we call this the *Share Update.* Both the FIXED-SHARE and VARIABLE-SHARE Algorithm first do the Loss Update followed by a Share Update, which differs for each algorithm. In a Share Update, a fraction of each experts' weight is added to the weight of each other expert. In the FIXED-SHARE Algorithm the experts share a fixed fraction of their weights with each other. This guarantees that the ratio of the weight of any expert to the total weight of all the experts may be bounded from below. Different forms of lower bounding the weights have been used by the WML algorithm and in the companion paper for learning shifting disjunctions (Auer & Warmuth, 1998) that appears in this journal issue. The latter two methods have been applied to learning problems where the loss is the discrete loss (i.e., counting mistakes). In contrast our methods work for the same general class of continuous loss functions that the STATIC-EXPERT algorithms can handle (Vovk, 1998; Haussler et al., 1998). This class includes all common loss functions such as the square loss, the relative entropy loss, and the hellinger loss. For this class there are tight bounds on the additional loss (Haussler et al., 1998) of the algorithm over the loss of the best expert (i.e., the non-shifting case). The FIXED-SHARE Algorithm obtains the additional loss of $O(c[(k+1)\log n + k \log \frac{\ell}{k} + k])$, which is essentially the same as the sketched algorithm that uses the STATIC-EXPERT algorithm with exponentially many partition-experts. The salient feature of the FIXED-SHARE Algorithm is that it still uses $O(1)$ time per expert per trial. However, this algorithm's additional loss still depends on the length of the sequence. Our lower bounds give some partial evidence that this seems to be unavoidable for loss

functions for which the loss in a single trial can be unbounded (such as for the relative entropy loss). For the case when the loss in a particular trial is at most one (such as for the square loss), we develop a second algorithm called the VARIABLE-SHARE Algorithm. This algorithm obtains bounds on the additional loss that are independent of the length of the sequence. It also shares weights after the Loss Update; however, the amount each expert shares now is commensurate with the loss of the expert in the current trial. In particular, when an expert has no loss, it does not share any weight.

Both versions of our Share Update are trivial to implement and cost a constant amount of time for each of the n weights. Although the algorithms are easy to describe, proving the additional loss bounds takes some care. We believe that our techniques constitute a practical method for tracking the predictions of the best expert with provable worst-case additional loss bounds. The essential ingredient for our success in a non-stationary setting, seems to be an algorithm for the stationary setting with a multiplicative weight update whose loss bound grows logarithmically with the dimension of the problem. Besides Vovk's AGGREGATING ALGORITHM (Vovk, 1998) and the WEIGHTED MAJORITY Algorithm (Littlestone & Warmuth, 1994), which only use the Loss Update, and are the basis of this work, a number of such algorithms have been developed. Examples are algorithms for learning linear threshold functions (Littlestone, 1988; Littlestone, 1989), and algorithms whose additional loss bound over the loss of the best linear combination of experts or sigmoided linear combination of experts is bounded (Kivinen & Warmuth, 1997; Helmbold, Kivinen & Warmuth, 1995). Significant progress has recently been achieved for other non-stationary settings building on the techniques developed in this paper (see discussion in the Conclusion Section).

The paper is outlined as follows. After some preliminaries (Section 2), we present the algorithms (Section 3), and give the basic proof techniques (Section 4). Sections 5 and 6 contain the detailed proofs for the FIXED-SHARE and VARIABLE-SHARE algorithms, respectively. The absolute loss is treated as a special case in Section 7. Section 8 discusses a subtle but powerful generalization of the VARIABLE-SHARE Algorithm, called the PROXIMITY-VARIABLE-SHARE Algorithm. The generalization leads to improved bounds for the case when best expert of the next segment is always likely to be "close" to the previous expert. Some preliminary lower bounds are given in Section 9. Simulation results on artificial data that exemplify our methods are given in Section 10. Finally, in Section 11 we conclude with a discussion of recent work. The casual reader who might not be interested in the detailed proofs is recommended to read the sections containing the preliminaries (Section 2), the algorithms (Section 3) and the simulations (Section 10).

2. Preliminaries

Let ℓ denote the number of trials and n denote the number of experts labeled $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_n$. When convenient we simply refer to an expert by its index; thus "expert i" refers to expert $\mathcal{E}_i$. The prediction of all n experts in trial t is referred to by the *prediction* tuple $\mathbf{x}_t$, while the prediction of expert i on trial t is denoted by $x_{t,i}$. These experts may be viewed as oracles external to the algorithm, and thus may represent the predictions of a neural net, a decision tree, a physical sensor or perhaps even of a human expert. The outcome of a trial t is y_t, while the prediction of the algorithm in trial t is $\hat{y}_t$. The instance-outcome pair

$(\mathbf{x}_t, y_t)$ is called the t-th *example*. In this paper the outcomes, the expert predictions and the predictions of the algorithm are all in $[0,1]$. Throughout this paper S always denotes an arbitrary sequence of examples, i.e., any sequence of elements from $[0,1]^n \times [0,1]$ of any length ℓ. A loss function $L(p,q)$ is a function $L : [0,1] \times [0,1] \to [0,\infty]$. We consider four loss functions in this paper: the square, the relative entropy, the hellinger, and the absolute loss:

$$
\begin{aligned}
&L_{\mathrm{sq}}(p,q) = (p-q)^2,\\
&L_{\mathrm{ent}}(p,q) = p \ln \tfrac{p}{q} + (1-p) \ln \tfrac{1-p}{1-q},\\
&L_{\mathrm{hel}}(p,q) = \tfrac{1}{2}((\sqrt{1-p} - \sqrt{1-q})^2 + (\sqrt{p} - \sqrt{q})^2) \text{ and}\\
&L_{\mathrm{abs}}(p,q) = |p-q|.
\end{aligned}
$$

On trial t the loss of the algorithm A is $L(y_t, \hat{y}_t)$. Similarly, the loss of expert i on trial t is $L(y_t, x_{t,i})$. We call a subsequence of contiguous trials a *segment*. The notation $[t..t']$ for non-negative integers $t \le t'$ denotes a *segment* starting on trial number t and ending on the trial t'. Rounded parens are used if the ending trial is not included in the segment. For the current sequence S we abbreviate the loss of expert i on the segment $[t..t')$ by $L([t..t'), i) = \sum_{s=t}^{t'-1} L(y_s, x_{s,i})$. The loss of the algorithm A over the whole trial sequence S is defined as $L(S,A) = L([1..\ell], A) = \sum_{t=1}^{\ell} L(y_t, \hat{y}_t)$.

We are now ready to give the main definition of this paper that is used for scenarios in which the best expert changes over time. Informally a *k-partition* slices a sequence into $k+1$ segments with an expert being associated with each segment. Formally, a *k-partition*, denoted by $\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)$, consists of three positive integers ℓ, n, k, and two tuples $\mathbf{t}$ and $\mathbf{e}$ of positive integers. The number ℓ is the length of the trial sequence S, n is the size of the expert pool, and k is number of *target shifts* ($k < \ell$). The tuple $\mathbf{t}$ has k elements $(t_1, \ldots, t_k)$ such that $1 < t_i \le \ell$ and $t_i < t_{i+1}$. Each t_i refers to one of the ℓ trials, and by convention we use $t_0 = 1, t_{k+1} = \ell + 1$. The tuple $\mathbf{t}$ divides the trial sequence S into $k+1$ *segments*, $[t_0..t_1), [t_1..t_2), \ldots, [t_k..t_{k+1})$. The segment $[t_i..t_{i+1})$ is called the ith segment. The 0th segment is also referred to as the initial segment. The tuple $\mathbf{e}$ has $k+1$ elements $(e_0, e_1, \ldots, e_k)$ such that $1 \le e_i \le n$ and $e_i \ne e_{i+1}$. The element e_i denotes the expert $\mathcal{E}_{e_i}$ which is *associated with* the ith segment $[t_i..t_{i+1})$. The loss of a given *k-partition* for loss function L and trial sequence S is

$$
L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)) = \sum_{i=0}^{k} L([t_i..t_{i+1}), e_i). \tag{1}
$$

3. The Algorithms

There are four algorithms considered in this paper – STATIC-EXPERT, FIXED-SHARE, VARIABLE-SHARE and PROXIMITY-VARIABLE-SHARE. The first three are summarized in Figure 1. The PROXIMITY-VARIABLE-SHARE Algorithm is a generalization of the VARIABLE-SHARE Algorithm; this algorithm is given in Figure 3. The discussion of this generalization is deferred to Section 8. For all algorithms the learning process proceeds in trials, where $t \ge 1$ denotes the trial number. The algorithms maintain one positive weight per expert. The weight $w^s_{t,i}$ (or its normalized version $v^s_{t,i}$) should be thought of as

Parameters: $0 < \eta, c$ and $0 \leq \alpha \leq 1$.

Initialization: Initialize the weights to $w^s_{1,1} = \ldots = w^s_{1,n} = 1/n$.

Prediction: Let $v_{t,i} = w^s_{t,i}/W_t$, where $W_t = \sum_{i=1}^n w^s_{t,i}$. Predict with

$$\hat{y}_t = \mathbf{pred}(\boldsymbol{v}_t, \boldsymbol{x}_t) \tag{2}$$

Loss Update: After receiving the tth outcome y_t,

$$\forall i : 1, \ldots, n : w^m_{t,i} = w^s_{t,i} e^{-\eta L(y_t, x_{t,i})}. \tag{3}$$

Share Updates of all three algorithms:

Static-expert

- $\forall i : 1, \ldots, n : w^s_{t+1,i} = w^m_{t,i}$ "no Share Update"

Fixed-share (4)

- $pool = \sum_{i=1}^n \alpha w^m_{t,i}$
- $\forall i : 1, \ldots, n : w^s_{t+1,i} = (1-\alpha) w^m_{t,i} + \frac{1}{n-1}\left(pool - \alpha w^m_{t,i}\right)$

Variable-share (5)

- $pool = \sum_{i=1}^n \left(1 - (1-\alpha)^{L(y_t, x_{t,i})}\right) w^m_{t,i}$
- $\forall i : 1, \ldots, n :$
  $w^s_{t+1,i} = (1-\alpha)^{L(y_t, x_{t,i})} w^m_{t,i} + \frac{1}{n-1}\left(pool - \left(1 - (1-\alpha)^{L(y_t, x_{t,i})}\right) w^m_{t,i}\right)$

Figure 1. The STATIC-EXPERT, FIXED-SHARE, and VARIABLE-SHARE algorithms.

a measurement of the algorithm's belief in the quality of the ith expert's predictions at the start of trial t. The weight of each expert is initialized to $1/n$.

The algorithms have the following three parameters: η, c and α. The parameter η is a learning rate quantifying how drastic the first update will be. The parameter c will be set to $1/\eta$ for most loss functions. (The absolute loss is an exception treated separately in Section 7.) The parameter α quantifies the rate of shifting that is expected to occur. The FIXED-SHARE Algorithm is designed for potentially unbounded loss functions, such as the relative entropy loss. The VARIABLE-SHARE Algorithm assumes that the loss per trial lies in $[0, 1]$. For the FIXED-SHARE Algorithm, α is the rate of shifting per trial. Thus, if five shifts are expected in a 1000 trial sequence, then $\alpha = 1/200$. For the VARIABLE-SHARE Algorithm, α is approximately the rate of shifting per unit of loss of the best partition. That is, if five shifts are expected to occur in a partition with a total loss of 80, then $\alpha \approx 1/16$. The tunings of the parameters η and c are considered in greater depth in Section 4, and for α in sections 5 and 6. Finally, the STATIC-EXPERT Algorithm does not use the parameter α since it assumes that no shifting occurs.

In each trial t the algorithm receives an instance summarizing the predictions of the n experts $\boldsymbol{x}_t$. The algorithm then plugs the current instance $\boldsymbol{x}_t$ and normalized weights

$\boldsymbol{v}_t$ into the prediction function $\mathbf{pred}(\boldsymbol{v}, \boldsymbol{x})$ in order to produce a prediction $\hat{y}_t$. In the simplest case, the algorithm predicts with the weighted mean of the experts' predictions, i.e., $\mathbf{pred}(\boldsymbol{v}, \boldsymbol{x}) = \boldsymbol{v} \cdot \boldsymbol{x}$. A more sophisticated prediction function introduced by Vovk (Vovk, 1998) will be discussed in Section 4. After predicting, the algorithm performs two update steps. The first update is the *Loss Update*; the second is the *Share Update*.

In the Loss Update the weight of expert i is multiplied by $e^{-\eta L_i}$, where L_i is the loss of the i-th expert in the current trial. Thus, no update occurs when $L_i = 0$. The learning rate η intensifies the effect of this update. We use $w_{t,i}^m$ to denote the weights in the middle of the two updates. These weights will be referred to as *intermediate* weights. The Share Update for the STATIC-EXPERT Algorithm is vacuous. However, for the other algorithms the Share Update is crucial. We briefly argue for the necessity of the share updates in the non-stationary setting, and then give an intuitive description of how they function.

When we move from predicting as well as the best expert to predicting as well as a sequence of experts, the Loss Update is no longer appropriate as the sole update. Assume we have two experts and two segments. In the first segment Expert 1 has small loss and Expert 2 a large loss. The roles are reversed for the second segment. By the end of the first segment, the Loss Update has caused the weight of Expert 2 to be almost zero. However, during the second segment the predictions of Expert 2 are important, and its weight needs to be recovered quickly. The share updates make sure that this is possible. The simulation in Section 10 furthers the intuition for why the share updates are needed. The two share updates are summarized below. A straightforward implementation costs $O(n)$ time per expert per trial:

$$\textbf{Fixed-share:} \quad w_{t,i}^s = (1-\alpha) w_{t,i}^m + \sum_{j \neq i}^{n} \frac{\alpha}{n-1} w_{t,j}^m, \tag{6}$$

$$\textbf{Variable-share:} \quad w_{t,i}^s = (1-\alpha)^{L(y_t, x_{t,i})} w_{t,i}^m + \sum_{j \neq i}^{n} \frac{\left(1-(1-\alpha)^{L(y_t, x_{t,j})}\right)}{n-1} w_{t,j}^m. \tag{7}$$

In contrast, the implementations in Figure 1, that use the intermediate variable "pool" cost $O(1)$ time per expert per trial. After the Loss Update, every expert "shares" a fraction of its weight equally with every other expert. The received weight enables an expert to recover its weights quickly relative to the other experts. In the Fixed-share Update (6) each expert shares a fraction of α of its weight in each trial. If one expert is perfect for a long segment, this type of sharing is not optimal, since the perfect expert keeps on sharing weight with possible non-perfect experts. The Variable-share Update (7) is more sophisticated: roughly, an expert shares weight when its loss is large. A perfect expert doesn't share, and if all other experts have high loss, it will eventually collect all the weight. However, when a perfect expert starts to incur high loss, it will rapidly begin to share its weight with the other experts, allowing a now good expert with previously small relative weight to recover quickly. As discussed above the parameter α is the shifting rate.

In the introduction we discussed an algorithm that uses exponentially many static experts, one for each partition. Our goal was to achieve bounds close to those of this inefficient algorithm by using only n weights. The bounds we obtain for our share algorithms are only

slightly weaker than the partition-expert algorithm and gracefully degrade when neither the length of the sequence ℓ nor the number of shifts k are known in advance.

4. Prediction Functions and Proof Techniques

We consider two choices of prediction functions. The simplest prediction is the weighted mean (Warmuth, 1997):

$$\mathbf{pred}_{\text{wmean}}(\boldsymbol{v}, \boldsymbol{x}) = \sum_{i=1}^{n} v_i x_i. \tag{8}$$

A more sophisticated prediction function giving slightly better bounds was introduced by Vovk (Vovk, 1998; Haussler et al., 1998). Define $L_0(z) = L(0, z)$ and $L_1(z) = L(1, z)$. Both of these functions must be monotone. Let $L_0^{-1}(z)$ and $L_1^{-1}(z)$ denote the inverses of $L_0(z)$ and $L_1(z)$. Vovk's prediction is now defined in two steps by

$$\begin{aligned} \Delta(y) &= -c \ln \textstyle\sum_{i=1}^{n} v_i e^{-\eta L(y, x_i)} \\ \mathbf{pred}_{\text{Vovk}}(\boldsymbol{v}, \boldsymbol{x}) &= \tfrac{L_0^{-1}(\Delta(0)) + L_1^{-1}(\Delta(1))}{2}. \end{aligned} \tag{9}$$

The following definition is a technical condition on the relation between the prediction function $\mathbf{pred}(\boldsymbol{v}, \boldsymbol{x})$, the loss function L, and the constants c and η.

DEFINITION 1 (HAUSSLER ET AL., 1998; VOVK, 1998) *A loss function L and prediction function* **pred** *are (c, η)-realizable for the constants c and η if*

$$L(\mathbf{pred}(\boldsymbol{v}, \boldsymbol{x}), y) \leq -c \ln \sum_{i=1}^{n} v_i e^{-\eta L(y, x_i)}, \tag{10}$$

for all $n \in \mathbf{Z}^+$, all examples $(\mathbf{x}, y) \in [0,1]^n \times [0,1]$, and all weight tuples $\mathbf{v} \in [0,1]^n$ of total weight 1.

We consider four loss functions in this paper: the square, the relative entropy, the hellinger, and the absolute loss (see Section 2). However, the algorithms are not limited to these loss functions. The techniques in (Vovk, 1998; Haussler et al., 1998; Warmuth, 1997) can determine the constants c and η for a wide class of loss functions. The algorithm is also easy to adapt for classification by using the majority vote (Littlestone & Warmuth, 1994) for the prediction function, and counting mistakes for the loss. In a practical application, no worst-case loss bounds may be provable for the given loss function. However, the share updates may still be useful. For an interesting application to the prediction of disk idle time see the work of Helmbold et al. (Helmbold, Long & Sherrod, 1996).

The square, relative entropy and hellinger losses are (c, η)-realizable for both $\mathbf{pred}_{\text{wmean}}$ and $\mathbf{pred}_{\text{Vovk}}$ with $(\eta = 1/c)$. The values of c (and hence of η) for the two prediction functions are summarized in Figure 2. Since the absolute loss has more complex bounds, we treat it in a section of its own. A smaller value of c leads to a smaller loss bound (see Lemma 1). The c values for $\mathbf{pred}_{\text{Vovk}}$ (cf. column two of Figure 2) are optimal for a large class of loss functions (Haussler et al., 1998).

Loss Functions:	c values: ($\eta = 1/c$)	
	$\mathbf{pred}_{\mathrm{wmean}}(\boldsymbol{v}, \boldsymbol{x})$	$\mathbf{pred}_{\mathrm{Vovk}}(\boldsymbol{v}, \boldsymbol{x})$
$L_{\mathrm{sq}}(p, q)$	2	$1/2$
$L_{\mathrm{ent}}(p, q)$	1	1
$L_{\mathrm{hel}}(p, q)$	1	$1/\sqrt{2}$

Figure 2. $(c, 1/c)$-realizability: c values for loss and prediction function pairings.

The proof of the loss bounds for each of the algorithms is based on the following lemma. The lemma embodies a key feature of the algorithms: the prediction is done such that the loss incurred by the algorithm is tempered by a corresponding change in total weight. This lemma gives the same inequality as the lemmas used in (Vovk, 1998; Haussler et al., 1998). The proof here is essentially the same, since the share updates do not change the total weight $W_t = \sum_{i=1}^{n} w_{t,i}^{s}$.

LEMMA 1 (VOVK, 1998; HAUSSLER ET AL., 1998) *For any sequence of examples S and for any expert i, the total loss of the master algorithms in Figure 1 may be bounded by*

$$L(S, A) \leq -c \ln w_{\ell+1,i}^{s}, \tag{11}$$

when the loss function L and prediction function **pred** *is (c, η)-realizable (cf. Definition 1 and Figure 2).*

Proof: Since L and **pred** are (c, η)-realizable, we have by Definition 1 that

$$L(y_t, \mathbf{pred}(\boldsymbol{v}_t, \boldsymbol{x}_t)) \leq -c \ln \sum_{i=1}^{n} v_{t,i} e^{-\eta L(y_t, x_{t,i})} = -c \ln \frac{1}{W_t} \sum_{i=1}^{n} w_{t,i}^{m}. \tag{12}$$

Since the share updates do not change, the total weight $\sum_{i=1}^{n} w_{t,i}^{m}$ is $\sum_{i=1}^{n} w_{t+1,i}^{s} = W_{t+1}$. This implies that

$$L(y_t, \hat{y}_t) \leq -c \ln \frac{W_{t+1}}{W_t}.$$

Hence, since $W_1 = 1$,

$$L(S, A) = \sum_{t=1}^{\ell} L(y_t, \hat{y}_t) \leq -c \ln W_{\ell+1} \leq -c \ln w_{\ell+1,i}.$$

■

So far we have used the same basic technique as in (Littlestone & Warmuth, 1994; Vovk, 1995; Cesa-Bianchi et al., 1997; Haussler et al., 1998), i.e., $c \ln W_t$ becomes the potential function in an amortized analysis. In the static expert case (when $\eta = 1/c$) the final weights have the form $w_{\ell+1,i}^{s} = e^{-L(S,\mathcal{E}_i)/c}/n$. Thus the above lemma leads to the bound

$$L(S, A) \leq L(S, \mathcal{E}_i) + c \ln n,$$

relating the loss of the algorithm to the loss of any static expert.

The share updates make it much more difficult to lower bound the final weights. Intuitively, there has to be sufficient sharing so that the weights can recover quickly. However, there should not be too much sharing, so that the final weights are not too low. In the following sections we bound final weights of individual experts in terms of the loss of a partition. The loss of any partition $(L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)))$ is just the sum of the sequence of losses defined by the sequence of experts in the partition. When an expert accumulates loss over a segment, we bound its weight using Lemma 2 for the FIXED-SHARE Algorithm and Lemma 7 for the VARIABLE-SHARE Algorithm. Since a partition is composed of distinct segments, we must also quantify how the weight is transferred from the expert associated with a segment to the expert associated with the following segment; this is done with Lemma 3 for the FIXED-SHARE Algorithm and Lemma 8 for the VARIABLE-SHARE Algorithm. The lower bounds on the weights are then combined with Lemma 1 to bound the total loss of the FIXED-SHARE Algorithm (Theorem 1) and the VARIABLE-SHARE Algorithm (Theorem 2).

5. Fixed-share Analysis

This algorithm works for unbounded loss functions, but its total additional loss grows with the length of the sequence.

LEMMA 2 *For any sequence of examples S the intermediate weight of expert i on trial t' is at least $e^{-\eta L([t..t'],i)}(1-\alpha)^{(t'-t)}$ times the weight of expert i at the start of trial t, where $t \leq t'$. Formally we have*

$$\frac{w^m_{t',i}}{w^s_{t,i}} \geq e^{-\eta L([t..t'],i)}(1-\alpha)^{(t'-t)}. \tag{13}$$

Proof: The combined Loss and Fixed-share Update (Equation (6)) can be rewritten as

$$w^s_{t+1,i} = \frac{\alpha}{n-1}\sum_{j\neq i}^{n} w^m_{t,j} + (1-\alpha)e^{-\eta L(y_t,x_{t,i})}w^s_{t,i}.$$

Then if we drop the additive term produced by the Share Update, we have

$$w^s_{t+1,i} \geq (1-\alpha)e^{-\eta L(y_t,x_{t,i})}w^s_{t,i}.$$

We apply the above iteratively on the trials $[t..t')$. Since we are bounding $w^m_{t',i}$ (the weights in trial t' after the Loss Update), the weight on trial t' is only reduced by a factor of $e^{-\eta L(y_{t'},x_{t',i})}$. Therefore we have

$$w^m_{t',i} \geq w^s_{t,i}\prod_{r=t}^{t'-1}\left[e^{-\eta L(y_r,x_{r,i})}(1-\alpha)\right]e^{-\eta L(y_{t'},x_{t',i})}.$$

By simple algebra and the definition of $L([a..b],i)$ the bound of the lemma follows. ∎

LEMMA 3 *For any sequence of examples S, the weight of an expert i at the start of trial $t+1$ is at least $\frac{\alpha}{n-1}$ times the intermediate weight of any other expert j on trial t.*

$$\frac{w^s_{t+1,i}}{w^m_{t,j}} \geq \frac{\alpha}{n-1}, \quad i \neq j \tag{14}$$

Proof: Expanding the Fixed-share Update (4) we have

$$w^s_{t+1,i} = (1-\alpha)w^m_{t,i} + \frac{\alpha}{n-1}\sum_{j\neq i}^{n} w^m_{t,j}.$$

Thus $w^s_{t+1,i} \geq \left(\frac{\alpha}{n-1}\right) w^m_{t,j}$, when $i \neq j$ and we are done. ∎

We can now bound the additional loss.

THEOREM 1 *Let S be any sequence of examples and let L and* **pred** *be (c,η)-realizable. Then for any k-shift sequence partition $\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)$ the total loss of the* FIXED-SHARE *Algorithm with parameter α satisfies*

$$L(S,A) \leq c\ln n + c\eta L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)) + c(\ell-k-1)\ln\frac{1}{1-\alpha} + ck[\ln\frac{1}{\alpha} + \ln(n-1)]. \tag{15}$$

Proof: Recall that e_k is the expert of the last segment. By Lemma 1, with $i = e_k$, we have

$$L(S,A) \leq -c\ln w^s_{\ell+1,e_k}. \tag{16}$$

We bound $w^s_{\ell+1,e_k}$ by noting that it "follows" the weight in an arbitrary partition. This is expressed in the following telescoping product:

$$w^s_{\ell+1,e_k} = w^s_{t_0,e_0}\frac{w^m_{t_1-1,e_0}}{w^s_{t_0,e_0}}\prod_{i=1}^{k}\left(\frac{w^s_{t_i,e_i}}{w^m_{t_i-1,e_{i-1}}}\cdot\frac{w^m_{t_{i+1}-1,e_i}}{w^s_{t_i,e_i}}\right)\frac{w^s_{\ell+1,e_k}}{w^m_{t_{k+1}-1,e_k}}.$$

Thus, applying lemmas 3 and 2, we have

$$w^s_{\ell+1,e_k} \geq w^s_{t_0,e_0}\prod_{i=0}^{k}\left[e^{-\eta L([t_i..t_{i+1}),e_i)}(1-\alpha)^{(t_{i+1}-t_i)-1}\right]\left(\frac{\alpha}{n-1}\right)^k\frac{w^s_{\ell+1,e_k}}{w^m_{t_{k+1}-1,e_k}}.$$

The final term $\frac{w^s_{\ell+1,e_k}}{w^m_{t_{k+1}-1,e_k}}$ equals one, since we do not apply the Share Update on the final trial; therefore by the definition of $L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S))$, we have

$$w^s_{\ell+1,e_k} \geq \frac{1}{n}e^{-\eta L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S))}(1-\alpha)^{\ell-k-1}\left(\frac{\alpha}{n-1}\right)^k.$$

We then substitute the above bound on $w^s_{\ell+1,e_k}$ into (16) and simplify to obtain (15). ∎

The bound of Theorem 1 holds for all k, and there is a tradeoff between the terms $ck \ln n$ and $c\eta L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S))$; i.e., when k is small the $ck \ln n$ term is small and the $c\eta L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S))$ term is large, and vice-versa. The optimal choice of α (obtained by differentiating the bound of Theorem 1) is $\alpha^* = \frac{k}{\ell-1}$. The following corollary rewrites the bound of Theorem 1 in terms of the optimal parameter choice α^*. The corollary gives an interpretation of the theorem's bound in terms of code length. We introduce the following notation. Let $H(p) = p \ln \frac{1}{p} + (1-p) \ln \frac{1}{1-p}$ be the binary entropy measured in nats, and $D(p\|q) = p \ln \frac{p}{q} + (1-p) \ln \frac{1-p}{1-q}$ be the binary relative entropy in nats.[2]

COROLLARY 1 *Let S be any sequence of examples and let L and* **pred** *be (c, η)-realizable. Then for any k-shift sequence partition $\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)$ the total loss of the* FIXED-SHARE *Algorithm with parameter α satisfies*

$$L(S, A) \le c \ln n + c\eta L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)) + c(\ell-1)\left[H(\alpha^*) + D(\alpha^*\|\alpha)\right] + ck \ln(n-1), \tag{17}$$

where $\alpha^* = \frac{k}{\ell-1}$. When $\alpha = \frac{k}{\ell-1}$, then this bound becomes

$$\begin{aligned} L(S, A) \le c \ln n + c\eta L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)) &+ ck(\ln \tfrac{\ell-1}{k} + \ln(n-1)) \\ &+ c(\ell-1-k) \ln(1 + \tfrac{k}{\ell-1-k}). \end{aligned} \tag{18}$$

For the interpretation of the bound we ignore the constants c, η and the difference between nats and bits. The terms $\ln n$ and $k \ln(n-1)$ account for encoding the $k+1$ experts of the partition: $\log n$ bits for the initial expert and $\log(n-1)$ bits for each expert thereafter. Finally, we need to encode where the k shifts occur (the inner boundaries of the partition). If α^* is interpreted as the probability that a shift occurs on any of the $\ell - 1$ trials, then the term $(\ell-1)\left[H(\alpha^*) + D(\alpha^*\|\alpha)\right]$ corresponds to the expected optimal code length (see Chapter 5 of (Cover & Thomas, 1991)) if we code the shifts with the estimate α instead of the true probability α^*. This bound is thus an example of the close similarity between prediction and coding as brought out by many papers (e.g., (Feder, Merhav & Gutman, 1992)).

Note that the α that minimizes the bound of Theorem 1 depends on k and ℓ which are unknown to the learner. In practice a good choice of α may be determined experimentally. However, if we have an upper bound on ℓ and a lower bound on k we may tune α in terms of these bounds.

COROLLARY 2 *Let S be any sequence of examples and $\hat{\ell}$ and $\hat{k}$ be any positive integers such that $\hat{k} < \hat{\ell} - 1$. Then by setting $\alpha = \hat{k}/(\hat{\ell}-1)$, the loss of the* FIXED-SHARE *Algorithm can be bounded by*

$$L(S, A) \le c \ln n + c\eta L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)) + ck(\ln \frac{\hat{\ell}-1}{\hat{k}} + \ln(n-1)) + c\hat{k}, \tag{19}$$

where $\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)$ is any partition of S such that $\ell \le \hat{\ell}$ and $k \ge \hat{k}$.

Proof: Recall the loss bound given in Theorem 1. By setting $\alpha = \frac{\hat{k}}{\hat{\ell}-1}$, we have

$$L(S,A) \le c\ln n + c\eta L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)) + ck[\ln \tfrac{\hat{\ell}-1}{\hat{k}} + \ln(n-1)] + c(\ell-1-k)\ln(1+\tfrac{\hat{k}}{\hat{\ell}-1-\hat{k}}). \tag{20}$$

We now separate out the term $(\ell-1-k)\ln(1+\frac{\hat{k}}{\hat{\ell}-1-\hat{k}})$ and apply the inequality $\ln(1+x) \le x$:

$$(\ell-1-k)\ln\left(1+\frac{\hat{k}}{\hat{\ell}-1-\hat{k}}\right) \le \hat{k}\frac{\ell-1-k}{\hat{\ell}-1-\hat{k}} = \hat{k}\left(1+\frac{(\ell-\hat{\ell})+(\hat{k}-k)}{\hat{\ell}-1-\hat{k}}\right) \le \hat{k}.$$

The last inequality follows from the condition that $\ell \le \hat{\ell}$ and $\hat{k} \le k$. We obtain the bound of the corollary by replacing $(\ell-1-k)\ln(1+\frac{\hat{k}}{\hat{\ell}-1-\hat{k}})$ in Equation (20) by its upper bound $\hat{k}$. [3]

■

6. Variable-share analysis

The VARIABLE-SHARE algorithm assumes that the loss of each expert per trial lies in $[0,1]$. Hence the VARIABLE-SHARE Algorithm works in combination with the square, hellinger, or absolute loss functions but not with the relative entropy loss function. The VARIABLE-SHARE Algorithm has an upper bound on the additional loss of the algorithm which is independent of the length of the trial sequence. We will abbreviate $w^s_{t,i}$ with $w_{t,i}$, since in this section we will not need to refer to the weight of an expert in the middle of a trial. We first give two technical lemmas that follow from convexity in r of β^r.

LEMMA 4 *If $\beta > 0$ and $r \in [0,1]$, then $\beta^r \le 1-(1-\beta)r$ and $1-(1-\beta)^r \ge \beta r$.*

LEMMA 5 *Given $b, c \in [0,1)$, $d \in (0,1]$ and $c+d \ge 1$, then $b^c(c+db^d) \ge b$.*

Proof: Since $d \ge 1-c$ and $b^d \ge b$, we have $db^d \ge (1-c)b$. Therefore, $c+db^d \ge c+(1-c)b = 1-(1-b)(1-c)$. Applying the first inequality of Lemma 4 to the RHS we have $c+db^d \ge b^{1-c}$, and thus

$$b^c(c+db^d) \ge b.$$

■

LEMMA 6 *At the beginning of trial $t+1$, we may lower bound the weight of expert i by either Expression (a) or Expression (b), where j is any expert different from i:*

$$w_{t+1,i} \ge \begin{cases} w_{t,i}e^{-\eta L(y_t,x_{t,i})}(1-\alpha)^{L(y_t,x_{t,i})} & \textbf{(a)} \\ w_{t,j}e^{-\eta L(y_t,x_{t,j})}\frac{\alpha}{n-1}L(y_t,x_{t,j}) & \textbf{(b)}. \end{cases}$$

Proof: Expanding the Loss Update and the Variable-share Update for a trial (cf. (7)) we have

$$w_{t+1,i} = w_{t,i}e^{-\eta L(y_t,x_{t,i})}(1-\alpha)^{L(y_t,x_{t,i})} + \tfrac{1}{n-1}\textstyle\sum_{j\ne i}^{n} w_{t,j}e^{-\eta L(y_t,x_{t,j})}\left(1-(1-\alpha)^{L(y_t,x_{t,j})}\right).$$

Expression (a) is obtained by dropping the summation term. For Expression (b) we drop all but one summand of the second term: $w_{t+1,i} \geq w_{t,j} e^{-\eta L(y_t,x_{t,j})} \frac{1-(1-\alpha)^{L(y_t,x_{t,j})}}{n-1}$. We then apply Lemma 4 and obtain (b). ■

LEMMA 7 *The weight of expert i from the start of trial t to the start of trial t', where $t < t'$, is reduced by no more than a factor of $\left[e^{-\eta}(1-\alpha)\right]^{L([t..t'),i)}$, i.e.*

$$\frac{w_{t',i}}{w_{t,i}} \geq \left[e^{-\eta}(1-\alpha)\right]^{L([t..t'),i)}. \tag{21}$$

Proof: From Lemma 6(a), we have that on trial t the weight of expert i is reduced as follows: $\frac{w_{t+1,i}}{w_{t,i}} \geq e^{-\eta L(y_t,x_{t,i})}(1-\alpha)^{L(y_t,x_{t,i})}$. If we apply this iteratively on the trials $[t..t')$, we have

$$\frac{w_{t',i}}{w_{t,i}} \geq \prod_{r=t}^{t'-1} \left[e^{-\eta L(y_r,x_{r,i})}(1-\alpha)^{L(y_r,x_{r,i})}\right] = \left[e^{-\eta}(1-\alpha)\right]^{L([t..t'),i)}.$$

■

In Lemma 6(b) we lower bound the weight transferred from expert p to expert q in a single trial. In the next lemma we show how weight is transferred over a sequence of trials.

LEMMA 8 *For any distinct experts p and q, if $L([t..t'),p) < 1$ and $1 \leq L([t..t'],p) < 2$, then on trial $t'+1$ we may lower bound the weight of expert q by*

$$w_{t'+1,q} \geq w_{t,p}\left[\frac{\alpha}{n-1}e^{-\eta}(1-\alpha)\right]\left[e^{-\eta}(1-\alpha)\right]^{L([t..t'],q)}. \tag{22}$$

Proof: As expert p accumulates loss in trials $t..t'$, it transfers part of its weight to the other $n-1$ experts, specifically to expert q, via the Variable-share Update. Let a_i, for $t \leq i \leq t'$, denote the weight *transferred* by expert p to expert q in trial i. Let $A = \sum_{i=t}^{t'} a_i$ denote the total weight transferred from expert p to expert q in trials $[t..t']$. The transferred weight, however, is still reduced as a function of the loss of expert q in successive trials. By Lemma 7, the weight a_i added in trial i is reduced by a factor of $\left[e^{-\eta}(1-\alpha)\right]^{L((i..t'],q)}$ during trials $i+1$ to t'. Thus

$$w_{t'+1,q} \geq \sum_{i=t}^{t'} a_i \left[e^{-\eta}(1-\alpha)\right]^{L((i..t'],q)}.$$

We lower bound each factor $\left[e^{-\eta}(1-\alpha)\right]^{L((i..t'],q)}$ by $\left[e^{-\eta}(1-\alpha)\right]^{L([t..t'],q)}$, and thus

$$w_{t'+1,q} \geq A\left[e^{-\eta}(1-\alpha)\right]^{L([t..t'],q)}. \tag{23}$$

To complete the proof of the lemma we still need to lower bound the total transferred weight A by $w_{t,p}\frac{\alpha}{n-1}e^{-\eta}(1-\alpha)$. Let l_i be the loss of expert p on trial i, i.e. $l_i = L(y_i, x_{i,p})$. From our assumption, we have $1 \leq \sum_{i=t}^{t'} l_i < 2$.

By direct application of Lemma 6(b), the weight a_t transferred by expert p to expert q in the first trial t of the segment is at least $w_{t,p}\frac{\alpha}{n-1}l_t e^{-\eta l_t}$. Likewise, we apply Lemma 7 over trials $[t..i)$ to expert p, and then apply Lemma 6(b) on trial i. This gives us a lower bound for the transferred weights a_i and the total transferred weight A:

$$a_i \geq w_{t,p}\frac{\alpha}{n-1}l_i e^{-\eta\sum_{j=t}^{i} l_j}(1-\alpha)^{\sum_{j=t}^{i-1} l_j},$$

$$A = \sum_{i=t}^{t'} a_i \geq w_{t,p}\frac{\alpha}{n-1}\sum_{i=t}^{t'} l_i e^{-\eta\sum_{j=t}^{i} l_j}(1-\alpha)^{\sum_{j=t}^{i-1} l_j}.$$

We split the last sum into two terms:

$$A \geq w_{t,p}\frac{\alpha}{n-1}\sum_{i=t}^{t'-1}\left(l_i e^{-\eta\sum_{j=t}^{i} l_j}(1-\alpha)^{\sum_{j=t}^{i-1} l_j}\right) + w_{t,p}\frac{\alpha}{n-1}l_{t'}e^{-\eta\sum_{j=t}^{t'} l_j}.$$

We upper bound all exponents of $(1-\alpha)$ by one; we also replace the sum in the first exponent by its upper bound, $\sum_{i=t}^{t'-1} l_i$. The substitutions $b = e^{-\eta}$, $c = \sum_{i=t}^{t'-1} l_i < 1$, and $d = l_{t'} \leq 1$, then lead to an application of Lemma 5. Thus we rewrite the above inequality as

$$A \geq w_{t,p}\frac{\alpha}{n-1}\left[cb^c(1-\alpha) + db^{c+d}(1-\alpha)\right] = w_{t,p}\frac{\alpha}{n-1}(1-\alpha)b^c\left(c + db^d\right),$$

and then apply Lemma 5. This gives us

$$A \geq w_{t,p}\frac{\alpha}{n-1}e^{-\eta}(1-\alpha).$$

■

The proof of the loss bound for the VARIABLE-SHARE Algorithm proceeds analogously to the proof of the FIXED-SHARE Algorithm's loss bound. In both cases we "follow" the weight of a sequence of experts along the sequence of segments. Within a segment we bound the weight reduction of an expert with Lemma 2 for the Fixed-share analysis and Lemma 7 for Variable-share analysis.

When we pass from one segment to the next, we bound the weight of the expert corresponding to the new segment by the weight of the expert in the former segment with lemmas 3 and 8, respectively. The former lemma used for the FIXED-SHARE Algorithm is very simple, since in each trial each expert always shared a fixed fraction of its weight. However, since the weight was shared on every trial, this produced a bound dependent on sequence length. In the VARIABLE-SHARE Algorithm we produce a bound independent of the length. This is accomplished by each expert sharing weight in accordance to its loss. However, if an expert does not accumulate significant loss, then we cannot use Lemma 8 to bound the weight of the following expert in terms of the previous expert. Nevertheless, if the former expert does not make significant loss in the current segment, this implies that we may bound the current segment with the former expert by *collapsing* the segments together.

In other words, the collapsing of two consecutive segments $([t_{i-1}..t_i),\ [t_i..t_{i+1}))$, creates a single segment $([t_{i-1}..t_{i+1})$. which is associated with the expert of the first segment of the original two consecutive segments. We can do this for any segment; thus we determine our bound in terms of the related *collapsed* partition whose loss is not much worse.

LEMMA 9 *For any partition $\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)$ there exists a* collapsed *partition $\mathcal{P}_{\ell,n,k',\mathbf{t}',\mathbf{e}'}(S)$ such that for each segment (except the initial segment), the expert* associated *with the prior segment incurs at least one unit of loss, and the loss on the whole sequence of the collapsed partition exceeds the loss of the original partition by no more than $k-k'$, i.e., the following properties hold:*

$$\forall i : 1 \le i \le k',\ L([t'_i..t'_{i+1}), e'_{i-1}) \ge 1 \text{ and} \tag{24}$$

$$L(\mathcal{P}_{\ell,n,k',\mathbf{t}',\mathbf{e}'}(S)) \le L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)) + k - k'. \tag{25}$$

Proof: Recall that e_i is the expert *associated* with the ith segment, which is comprised of the trials $[t_i..t_{i+1})$. If in any segment i, the loss of the expert e_{i-1} associated with the prior segment $(i-1)$ is less than one, then we merge segment $i-1$ with segment i. This combined segment in the new partition is *associated* with expert e_{i-1}. Formally in each iteration, we decrement k by one, and we delete e_i and t_i from the tuples $\mathbf{e}$ and $\mathbf{t}$. We continue until (24) holds. We bound the loss of the collapsed partition $\mathcal{P}_{\ell,n,k',\mathbf{t}',\mathbf{e}'}(S)$, by noting that the loss of the new expert on the subsumed segment is at most one. Thus per application of the transformation, the loss increases by at most one. Thus since there are $k-k'$ applications, we are done. ■

THEOREM 2 [4] *Let S be any sequence of examples, let L and* **pred** *be (c,η)-realizable, and let L have a [0,1] range. Then for any partition $\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)$ the total loss of the* VARIABLE-SHARE *algorithm with parameter α satisfies*

$$L(S,A) \le c\ln n + c[\eta + \ln\frac{1}{1-\alpha}]L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)) + ck[\eta + \ln\frac{1}{\alpha} + \ln\frac{1}{1-\alpha} + \ln(n-1)].$$

Proof: By Lemma 1 with $i = e_k$ we have

$$L(S,A) \le -c\ln w^s_{\ell+1,e_k}. \tag{26}$$

Let $\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)$ be an arbitrary partition. For this proof we need the property that the loss in each segment (except the initial segment), with regard to the expert *associated* with the prior segment, is at least one (cf (24)). If this property does not hold, we use Lemma 9 to replace $\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)$ by a collapsed partition $\mathcal{P}_{\ell,n,k',\mathbf{t}',\mathbf{e}'}(S)$ for which the property does hold. If the property holds already for $\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)$, then for notational convenience we will refer to $\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)$ by $\mathcal{P}_{\ell,n,k',\mathbf{t}',\mathbf{e}'}(S)$. Recall that the loss of $\mathcal{P}_{\ell,n,k',\mathbf{t}',\mathbf{e}'}(S)$ exceeds the loss of $\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)$ by no more than $k-k'$.

Since (24) holds, there exists a trial q_i in the ith segment (for $1 \le i \le k'$) such that $L([t'_i..q_i), e'_{i-1}) < 1$ and $2 > L([t'_i..q_i], e'_{i-1}) \ge 1$. We now express $w_{\ell+1,e'_{k'}}$ as the telescoping product

$$w_{\ell+1,e'_{k'}} = w_{t'_0,e'_0}\frac{w_{t'_1,e'_0}}{w_{t'_0,e'_0}}\prod_{i=1}^{k'}\left(\frac{w_{q_i,e'_i}}{w_{t'_i,e'_{i-1}}}\cdot\frac{w_{t'_{i+1},e'_i}}{w_{q_i,e'_i}}\right).$$

Applying lemmas 7 and 8 we have

$$w_{\ell+1,e'_{k'}} \geq w_{t'_0,e'_0}[e^{-\eta}(1-\alpha)]^{L([t'_0..t'_1),e'_0)} \prod_{i=1}^{k'} \left(\left(\frac{\alpha}{n-1}\right) e^{-\eta}(1-\alpha)[e^{-\eta}(1-\alpha)]^{L([t'_i..t'_{i+1}),e'_i)} \right),$$

which simplifies to the following bound:

$$\begin{aligned} w_{\ell+1,e'_{k'}} &\geq [e^{-\eta}(1-\alpha)]^{L(\mathcal{P}_{\ell,n,k',\mathbf{t}',\mathbf{e}'}(S))+k'} \left(\frac{\alpha}{n-1}\right)^{k'} \\ &\geq [e^{-\eta}(1-\alpha)]^{L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S))+k} \left(\frac{\alpha}{n-1}\right)^{k}. \end{aligned}$$

The last inequality follows from (25). Thus if we substitute the above bound on $w_{\ell+1,e'_{k'}}$ into (26) and simplify, we obtain the bound of the theorem. ■

Again we cannot optimize the above upper bound as a function of α, since k and $L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S))$ are not known to the learning algorithm. Below we tune α based on an upper bound of $L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S))$. The same approach was used in Corollary 2.[5]

COROLLARY 3 *Let S be any sequence of examples and $\hat{L}$ and $\hat{k}$ be any positive reals. Then by setting $\alpha = \frac{\hat{k}}{2\hat{k}+\hat{L}}$, the loss of the* VARIABLE-SHARE *Algorithm can be bounded as follows:*

$$L(S,A) \leq c\ln n + c\eta L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)) + ck\left(\ln\left(\frac{\hat{L}}{\hat{k}}\right) + \ln(n-1) + \ln\frac{9}{2} + \eta\right) + c\hat{k}, \tag{27}$$

where $\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)$ is any partition such that $L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)) \leq \hat{L}$, and in addition $\hat{k} \leq \hat{L}$.

For any partition $\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)$ for which $L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)) \leq \hat{L}$ and $\hat{k} \geq \hat{L}$, we obtain the upper bound

$$L(S,A) \leq c\ln n + c\eta L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)) + ck\left(\ln(n-1) + \ln\frac{9}{2} + \eta\right) + \frac{1}{2}c\hat{k}. \tag{28}$$

Proof: We proceed by upper bounding the three terms containing α from the bound of Theorem 2 (we use $\alpha = \frac{\hat{k}}{2\hat{k}+\hat{L}}$):

$$L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S))\ln\left(\frac{2\hat{k}+\hat{L}}{\hat{L}+\hat{k}}\right) + k\left[\ln\left(\frac{2\hat{k}+\hat{L}}{\hat{L}+\hat{k}}\right) + \ln\left(\frac{2\hat{k}+\hat{L}}{\hat{k}}\right)\right]. \tag{29}$$

We rewrite the above as:

$$L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S))\ln(1+\frac{\hat{k}}{\hat{L}+\hat{k}}) + k\left[\ln\frac{(2\hat{k}+\hat{L})^2}{(\hat{k}+\hat{L})\hat{k}} + \ln\frac{\hat{L}}{\hat{k}}\right].$$

We apply the identity $\ln(1+x) \leq x$ and bound $L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S))$ by $\hat{L}$, giving the following upper bound of the previous expression:

$$\frac{\hat{L}\hat{k}}{\hat{L}+\hat{k}} + k \ln \frac{\hat{L}}{\hat{k}} + k \ln \frac{(2\hat{k}+\hat{L})^2}{(\hat{k}+\hat{L})\hat{L}}.$$

If $\hat{k} \leq \hat{L}$, then $\frac{\hat{L}\hat{k}}{\hat{L}+\hat{k}} < \hat{k}$. By simple calculus, $k \ln \frac{(2\hat{k}+\hat{L})^2}{(\hat{k}+\hat{L})\hat{L}} \leq k \ln \frac{9}{2}$, when $0 < \hat{k} \leq \hat{L}$. Therefore the above is upper bounded by

$$k \ln \frac{\hat{L}}{\hat{k}} + k \ln \frac{9}{2} + \hat{k}.$$

Using this expression to upper bound Equation (29), we obtain Equation (27).

When $\hat{k} \geq \hat{L}$, we upper bound Equation (29) by

$$\frac{\hat{L}\hat{k}}{\hat{L}+\hat{k}} + k \ln \frac{(2\hat{k}+\hat{L})^2}{(\hat{k}+\hat{L})\hat{k}}.$$

The first term is bounded by $\frac{1}{2}\hat{k}$. The second term $\frac{(2\hat{k}+\hat{L})^2}{(\hat{k}+\hat{L})\hat{k}}$ is at most $k \ln \frac{9}{2}$ in the region $0 < \hat{L} \leq \hat{k}$, and thus the above is upper bounded by

$$k \ln \frac{9}{2} + \frac{1}{2}\hat{k}.$$

We use the above expression to upper bound Equation (29). This gives us Equation (28) and we are done. ■

7. Absolute Loss Analysis

The absolute loss function $L_{abs}(p,q) = |p-q|$ is (c,η)-realizable with both the prediction functions $\mathbf{pred}_{\text{Vovk}}$ and $\mathbf{pred}_{\text{wmean}}$; however, $c\eta > 1$. Thus the tuning is more complex, and for the sake of simplicity we use the weighted mean prediction (Littlestone & Warmuth, 1994) in this section.

THEOREM 3 (LITTLESTONE & WARMUTH, 1994) *For $\eta \in [0,\infty)$, the absolute loss function $L_{abs}(p,q) = |p-q|$ is $(\frac{1}{1-e^{-\eta}},\eta)$-realizable for the prediction function $\mathbf{pred}_{wmean}(\boldsymbol{v},\boldsymbol{x})$.*

To obtain a slightly tighter bound we could also have used the Vee Algorithm for the absolute loss, which is $((2\ln\frac{2}{1+e^{-\eta}})^{-1},\eta)$-realizable (Haussler et al., 1998). This algorithm takes $O(n \log n)$ time to produce its prediction. Both the weighted mean and the Vee prediction allow the outcomes to lie in $[0,1]$. For binary outcomes with the absolute loss, $O(n)$ time prediction functions exist with the same realizability criterion as the Vee prediction (Vovk, 1998; Cesa-Bianchi et al., 1997).

Unlike the $(c,1/c)$-realizable loss functions discussed earlier (cf. Figure 2), the absolute value loss does not have constant parameters, and thus it must be tuned. In practice, the tuning of η may be produced by numerical minimization of the upper bounds. However, we use a tuning of η produced by Freund & Schapire (1997).

THEOREM 4 (LEMMA 4 (FREUND & SCHAPIRE, 1997)) *Suppose* $0 \le P \le \hat{P}$ *and* $0 < Q \le \hat{Q}$. *Let* $\eta = g(\hat{P}/\hat{Q})$, *where* $g(z) = \ln(1 + \sqrt{2/z})$; *then*

$$\frac{P\eta + Q}{1 - e^{-\eta}} \le P + \sqrt{2\hat{P}\hat{Q}} + Q.$$

We now use the above tuning in the bound for the VARIABLE-SHARE Algorithm (Theorem 2).

THEOREM 5 *Let the loss function be the absolute loss. Let* S *be any sequence of examples, and* $\hat{L}$ *and* $\hat{k}$ *be any positive reals such that* $k \le \hat{k}$, $L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)) \le \hat{L}$, *and* $k \le \hat{L}$. *Set the two parameters of the* VARIABLE-SHARE *algorithm* α *and* η *to* $\frac{\hat{k}}{2\hat{k}+\hat{L}}$ *and* $\ln(1 + \sqrt{2(\hat{Q}/\hat{P})})$, *respectively, where*

$$\hat{P} = \hat{L} + \hat{k} and \hat{Q} = \ln n + \hat{k}\left(\ln(n-1) + \ln(\frac{\hat{L}}{\hat{k}}) + \ln\frac{9}{2}\right) + \hat{k}.$$

Then the loss of the Algorithm with weighted mean prediction can be bounded as follows:

$$\begin{aligned} L(S,A) \;\le\; & L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)) + k \\ & + \sqrt{2\left(\hat{L}+\hat{k}\right)\left(\ln n + \hat{k}\left(\ln(n-1) + \ln(\frac{\hat{L}}{\hat{k}}) + \ln\frac{9}{2}\right) + \hat{k}\right)} \\ & + \ln n + k\left(\ln(n-1) + \ln(\frac{\hat{L}}{\hat{k}}) + \ln\frac{9}{2}\right) + \hat{k}. \end{aligned}$$

Alternatively, let $\hat{L}$ *and* $\hat{k}$ *be any positive reals such that* $k \le \hat{k}$, $L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)) \le \hat{L}$, *and* $k \ge \hat{L}$. *Set the two parameters of the* VARIABLE-SHARE *algorithm* α *and* η *to* $\frac{\hat{k}}{2\hat{k}+\hat{L}}$ *and* $\ln(1 + \sqrt{2(\hat{Q}/\hat{P})})$, *respectively, where*

$$\hat{P} = \hat{L} + \hat{k} and \hat{Q} = \ln n + \hat{k}\left(\ln(n-1) + \ln\frac{9}{2}\right) + \frac{1}{2}\hat{k}.$$

Then the loss of the Algorithm with weighted mean prediction can be bounded as follows:

$$\begin{aligned} L(S,A) \;\le\; & L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)) + k \\ & + \sqrt{2\left(\hat{L}+\hat{k}\right)\left(\ln n + \hat{k}\left(\ln(n-1) + \ln\frac{9}{2}\right) + \frac{1}{2}\hat{k}\right)} \\ & + \ln n + k\left(\ln(n-1) + \ln\frac{9}{2}\right) + \frac{1}{2}\hat{k}. \end{aligned}$$

Parameters: $0 < \eta, c$ and $0 \leq \alpha \leq 1$. $\forall i, j, k : 1, \ldots, n : 0 < \lambda_i^0, \lambda_{j,k}$, $\sum_{i=1}^n \lambda_i^0 = 1$, and $\forall j : 1, \ldots, n : \sum_{k \neq j}^n \lambda_{j,k} = 1$.

Initialization: Initialize the weights to $w_{1,1}^s = \lambda_1^0, \ldots, w_{1,n}^s = \lambda_n^0$.

Prediction: Let $v_{t,i} = w_{t,i}^s / W_t$, where $W_t = \sum_{i=1}^n w_{t,i}^s$. Predict with

$$\hat{y}_t = \mathbf{pred}(\boldsymbol{v}_t, \boldsymbol{x}_t)$$

Loss Update: After receiving the tth outcome y_t,

$$\forall i : 1, \ldots, n : w_{t,i}^m = w_{t,i}^s e^{-\eta L(y_t, x_{t,i})}.$$

Proximity-variable-share Update

- $\forall i : 1, \ldots, n :$

$$w_{t+1,i}^s = (1-\alpha)^{L(y_t, x_{t,i})} w_{t,i}^m + \sum_{j \neq i}^n \lambda_{j,i} \left(1 - (1-\alpha)^{L(y_t, x_{t,j})}\right) w_{t,j}^m$$

Figure 3. The PROXIMITY-VARIABLE-SHARE algorithm.

8. Proximity-variable-share Analysis

In this section we discuss the PROXIMITY-VARIABLE-SHARE Algorithm (see Figure 3). Recall that in the VARIABLE-SHARE Algorithm each expert shared a fraction of weight dependent on its loss in each trial; that fraction is then shared uniformly among the remaining $n-1$ experts. The PROXIMITY-VARIABLE-SHARE Algorithm enables each expert to share non-uniformly to the other $n-1$ experts. The Proximity-variable-share Update now costs $O(n)$ per expert per trial instead of $O(1)$ (see Figure 3). This algorithm allows us to model situations where we have some prior knowledge about likely pairs of consecutive experts.

Let us consider the parameters of the algorithm. The n-tuple $\lambda^{\mathbf{0}} \in [0,1]^n$ ($\sum_{i=1}^n \lambda_i^0 = 1$) contains the initial weights of the algorithm, i.e., $w_{1,i}^s = \lambda_i^0$. The second additional parameter besides η and c is a complete directed graph λ of size n without loops. The edge weight $\lambda_{j,k}$ is the fraction of the weight shared by expert j to expert k. Naturally, for any vertex, all outgoing edges must be nonnegative and sum to one. The $\lambda^{\mathbf{0}}$ probability distribution is a prior for the initial expert and the $\lambda_{\mathbf{j},\cdot}$ probability distribution is a prior for which expert will follow expert j. Below is the upper bound for the PROXIMITY-VARIABLE-SHARE Algorithm. The FIXED-SHARE Algorithm could be generalized similarly to take proximity into account.

THEOREM 6 *Let S be any sequence of examples, let L and* **pred** *be (c, η)-realizable, and let L have a [0,1] range. Then for any partition $\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)$, the total loss of the* PROXIMITY-VARIABLE-SHARE *Algorithm with parameter α satisfies*

$$L(S,A) \leq c[\eta + \ln\frac{1}{1-\alpha}] L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)) + ck[\eta + \ln\frac{1}{\alpha} + \ln\frac{1}{1-\alpha}] + \ln \lambda_{e_0}^0 + \sum_{i=1}^k \ln \lambda_{e_{i-1},e_i}. \tag{30}$$

Proof: We omit the proof of this bound since it is similar to the corresponding proof of Theorem 2 for the VARIABLE-SHARE Algorithm: The only change is that the $\frac{1}{n}$ and $\frac{1}{n-1}$ fractions are replaced by the corresponding λ parameters. ■

Note that setting $\lambda_{e_0} = \frac{1}{n}$ and $\lambda_{e_{i-1},e_i} = \frac{1}{n-1}$ gives the previous bound for the VARIABLE-SHARE Algorithm (Theorem 2). In that case the last sum is $O(k \ln n)$, accounting for the code length of the names of the best experts (except the first one). Using the PROXIMITY-VARIABLE-SHARE Algorithm we can get this last sum to $O(k)$ in some cases.

For a simple example, assume that the processors are on a circular list and that for the two processors of distance d from processor i, $\lambda_{i,i+d \mod n} = \lambda_{i,i-d \mod n} \propto 1/d^2$. Now if the next best expert is always at most a constant away from the previous one, then the last sum becomes $O(k)$. Of course, other notions of closeness and choices of the λ parameters might be suitable. Note that there is a price for decreasing the last sum: the update time is now $O(n^2)$ per trial. However, if for each expert i all arrows that end at i are labeled with the same value, then the Share Update of the PROXIMITY-VARIABLE-SHARE Algorithm is still $O(n)$.

9. Lower Bounds

The upper bounds for the FIXED-SHARE Algorithm grow with the length of the sequence. The additional loss of the algorithm over the loss of the best k-partition is approximately $(k+1)\ln n + k\ln(\ell/k)$. This holds for unbounded loss functions such as the relative entropy loss. When restricting the loss to lie in $[0,1]$, the VARIABLE-SHARE Algorithm gives an additional loss bound of approximately $(k+1)\ln n + k\ln(L/k)$, where L is the loss of the best k-partition and $k < L$. One natural question is whether a similar reduction is possible for unbounded loss functions. In other words, whether for an unbounded loss function a bound of the same form is possible with ℓ replaced by $\min\{\ell, L\}$. We give evidence to the contrary. We give an adversary argument that forces any algorithm to make $\ln 2 + \ln(\ell - 1)$ loss over the best one-partition (for which the adversary sets $L = L(\mathcal{P}_{\ell,2,1,\mathbf{t},\mathbf{e}}) = 0$). In this section we limit ourselves to giving this construction. It can easily be extended to an adversary that forces $\ln(n) + \ln(\ell - \log_2 n)$ additional loss over the best one-partition with n experts. By iterating the adversary, we may force

$$k\ \ [\ln(n-1) + \ln(\frac{\ell}{k} - \log_2(n-1))]$$

additional loss over the best k-partition. (Here we assume that $\log_2(n-1)$ and $\frac{\ell}{k}$ are positive integers, and $\frac{\ell}{k} > \log_2(n-1)$.)

THEOREM 7 *For the relative entropy loss there exists an example sequence S of length ℓ with two experts such that $L(\mathcal{P}_{\ell,2,1,\mathbf{t},\mathbf{e}}) = 0$, i.e., there is a partition with a single shift of loss 0, and furthermore, for any algorithm A,*

$$L(S,A) \geq \ln 2 + \ln(\ell - 1). \tag{31}$$

1. $\forall t : \mathbf{x}_t = (0, 1)$.
2. On trial $t = 0$, set y_0 to 0 if $\hat{y}_0 \geq \frac{1}{2}$ and to 1 otherwise.
   (Assume without loss of generality that $\hat{y}_0 \geq \frac{1}{2}$ and thus $y_0 = 0$).
3. New trial: $t = t + 1$.
4. If $\hat{y}_t > \frac{1}{\ell - t}$ then
   (A) $y_t = 0$. (Invariant conditions: 4(i) $L_t \geq \ln \frac{\ell - t}{\ell - t - 1}$, 4(ii) $\sum_{t'=0}^{t} L_{t'} \geq \ln 2 + \ln \frac{\ell - 1}{\ell - t - 1}$).
5. else
   (A) $y_t = 1$. (Invariant conditions: 5(i) $L_t \geq \ln(\ell - t)$, 5(ii) $\sum_{t'=0}^{t} L_{t'} \geq \ln 2 + \ln(\ell - 1)$).
   (B) Go to step 7.
6. If $t < \ell - 2$ then go to step 3.
7. Let $y_t = 1$ for the remaining trial(s) and exit.

Figure 4. Adversary's strategy.

Proof: The adversary's strategy is described in Figure 4. We use $\hat{y}_t$ to denote the prediction of an arbitrary learning algorithm, and $L_t = L_{ent}(y_t, \hat{y}_t)$ to denote the loss at trial t. For convenience we number the trials from $t = 0 \ldots \ell - 1$ instead of $t = 1 \ldots \ell$.

There are two experts; one always predicts 0 and the other always predicts 1. The adversary returns a sequence of 0 outcomes followed by a sequence of 1 outcomes such that neither sequence is empty. Thus, there is a single shift in the best partition, and this partition has loss 0.

We now prove that $L(S, A) \geq \ln 2 + \ln(\ell - 1)$, thus proving the lemma. Clearly $L_0 \geq \ln 2$ (see Step 2). Without loss of generality assume $\hat{y}_0 \geq \frac{1}{2}$. Note that the threshold for $\hat{y}_t$ is $\frac{1}{\ell - t}$. Furthermore, $L_{ent}(0, \frac{1}{\ell - t}) = \ln \frac{\ell - t}{\ell - t - 1}$ and $L_{ent}(1, \frac{1}{\ell - t}) = \ln(\ell - t)$. Thus, the conditions 4(i) and 5(i) follow. Condition 4(ii) holds by simple induction. If a shift occurs, then Condition 5(ii) holds, since by Condition 4(ii) in trial $t - 1$ we have that $\sum_{t'=0}^{t-1} L_{t'} \geq \ln 2 + \ln \frac{\ell - 1}{\ell - t}$. Therefore, when we add L_t, which is at least $\ln(\ell - t)$ by Condition 5(i), we obtain Condition 5(ii) and we are done. If Step 5 is never executed then the shift to $y_t = 1$ occurs in the last trial $\ell - 1$ since Step 6 is skipped. Thus, if Step 5 is never executed then $\sum_{t'=0}^{\ell-2} L_{t'} \geq \ln 2 + \ln(\ell - 1)$ in trial $t = \ell - 2$ (Condition 4(ii)), which is again the bound of the lemma. ■

We first reason that this lower bound is tight by showing that the upper bounds of the algorithms discussed in this paper are close to the lower bound. The number of partitions when $n = 2$ and $k = 1$ is $2(\ell - 1)$. Thus, we may expand the set of n experts into $2(\ell - 1)$

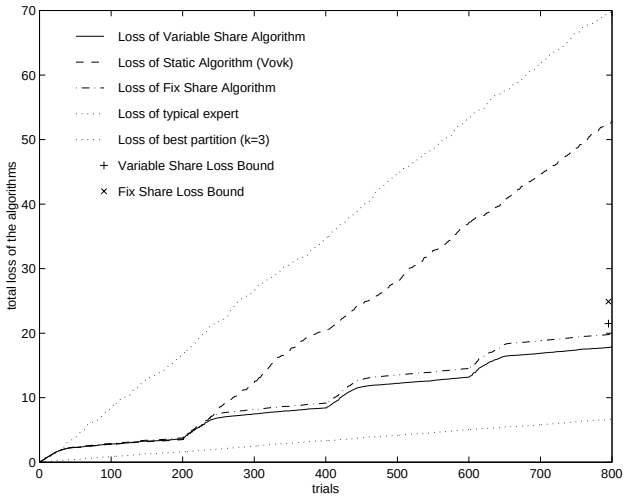

Figure 5. Loss of the VARIABLE-SHARE Algorithm vs the STATIC-EXPERT Algorithm.

partition-experts as discussed in the introduction. Using the STATIC-EXPERT Algorithm with the weighted mean prediction gives an upper bound of $\ln 2 + \ln(\ell - 1)$ on the total loss of the algorithm when the loss of the best partition is zero. This matches the above lower bound. Second, the bound of the FIXED-SHARE Algorithm (cf. Corollary 1) is larger than the lower bound by $(\ell - 2)\ln(1 + \frac{1}{\ell - 2})$, and this additional term may be upper bounded by 1.

10. Simulation Results

In this section we discuss some simulations on artificial data. These simulations are mainly meant to provide a visualization of how our algorithms track the predictions of the best expert and should not be seen as empirical evidence of the practical usefulness of the algorithms. We believe that the merits of our algorithms are more clearly reflected in the strong upper bounds we prove in the theorems of the earlier sections. Simulations only show the loss of an algorithm for a typical sequence of examples. The bounds of this paper are worst-case bounds that hold even for adversarially-generated sequences of examples. Surprisingly, the losses of the algorithms in the simulations with random sequences are very close to the corresponding worst-case bounds which we have proven in this paper. Thus, our simulations show that our loss bounds are tight for some sequences.

We compared the performance of the STATIC-EXPERT Algorithm to the two Share algorithms in the following setting. We chose to use the square loss as our loss function, because of its widespread use and because the task of tuning the learning rate for this loss function is simple. We used the Vovk prediction function (cf. Equation 9), and we chose $\eta = 2$ and $c = 1/2$ in accordance with Figure 2. We considered a sequence of 800 trials with four distinct *segments*, beginning at trials 1, 201, 401, and 601. On each trial the

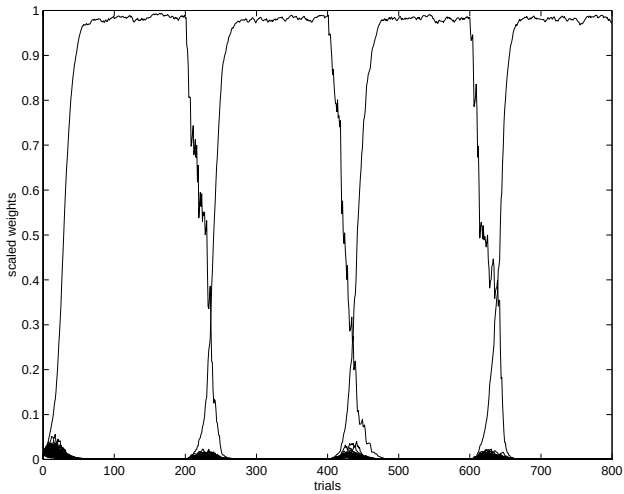

Figure 6. Relative Weights of the VARIABLE-SHARE Algorithm.

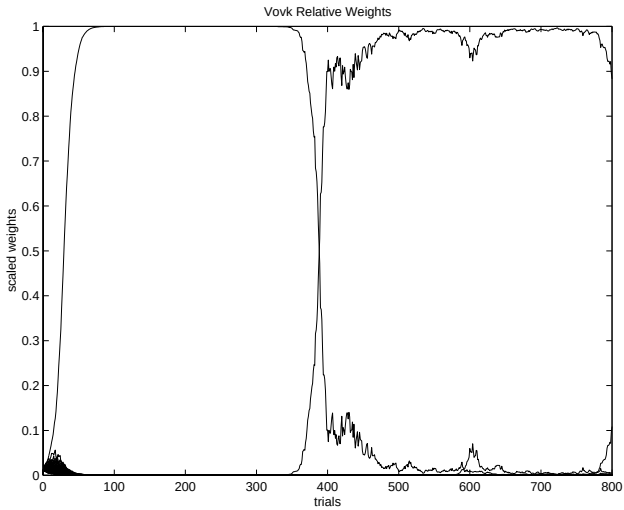

Figure 7. Relative Weights of the STATIC-EXPERT Algorithm.

outcome (y_t) was 0. The prediction tuple ($\mathbf{x}_t$) contained the predictions of 64 experts. When we generated the predictions of the 64 experts, we chose a different expert as the best one for each segment. The best experts always have an expected loss of $1/120$ per trial. The other 63 experts have an expected loss of $1/12$ per trial. At the end of each segment a new "best expert" was chosen. Since the *outcome* was always 0, we generated these expected losses by sampling predictions from a uniform random distribution on $(0, \frac{1}{2})$ and

$(0, \frac{1}{2}\sqrt{\frac{1}{10}})$ for the "typical" and "best" experts, respectively. Thus, the expected loss for the best[6] partition, denoted by the segment boundaries above, is $\frac{800}{120} = 6\frac{2}{3}$, with a variance of $\sigma^2 \approx .044$. The actual loss of the best partition in the particular simulation used for the plots was 6.47. For the FIXED-SHARE Algorithm we tuned α_f based on the values of $k = 3$ and $\ell = 800$ ($\alpha_f = \frac{3}{799} = 0.00375$), using the α_f tuning suggested in Corollary 1. For the VARIABLE-SHARE Algorithm we tuned α_v based on the values of $k = 3$ and $\hat{L} = 6.73$ ($\alpha_v = \frac{3}{6+6.73} = 0.24$), using the α_v tuning suggested in Corollary 3. Using theorems 1 and 2 we calculated a worst case upper bound on the loss of the FIXED-SHARE Algorithm and the VARIABLE-SHARE Algorithm of 24.89 and 21.50, respectively (see "$\times$" and "$+$" marks in Figure 5). The simulations on artificial data show that our worst-case bounds are rather tight even on this very simple artificial data.

There are many heuristics for finding a suitable tuning. We used the tunings prescribed by our theorem, but noticed that for these types of simulations the results are relatively insensitive to the tuning of α. For example, in calculating α_v for the VARIABLE-SHARE Algorithm when $\hat{L}$ was overestimated by 10 standard deviations, the loss bound for our algorithm increased by only 0.02, while the actual loss of the algorithm in the simulation increased by 0.17.

In Figure 5, we have plotted the loss of the STATIC-EXPERT Algorithm versus the loss of the two Share algorithms. Examination of the figure shows that on the first segment the STATIC-EXPERT Algorithm performed comparably to the Share algorithms. However, on the remaining three segments, the STATIC-EXPERT Algorithm performed poorly, in that its loss is essentially as bad as the loss of a "typical" expert (the slope of the total loss of a typical expert and the STATIC-EXPERT Algorithm is essentially the same for the later segments). The Share algorithms performed poorly at the beginning of a new segment; however, they quickly "learned" the new "best" expert for the current segment. The Share algorithms' loss plateaued to almost the same slope as the slope of the total loss of the best expert. The two Share algorithms had the same qualitative behavior, even though the FIXED-SHARE Algorithm incurred approximately 10% additional loss over the VARIABLE-SHARE Algorithm. In our simulations we tried learning rates η slightly smaller than two, and verified that even with other choices for the learning rates, the total loss of the STATIC-EXPERT algorithm does not improve significantly.

In Figures 6 and 7, we plotted the weights of the normalized weight vector $\mathbf{w_t}$ that is maintained by the VARIABLE-SHARE Algorithm and the STATIC-EXPERT Algorithm over the trial sequence. In Figure 6, we see that the VARIABLE-SHARE Algorithm shifts the relative weights rapidly. During the latter part of each segment, the relative weight of the best expert is almost one (the corresponding plot of the FIXED-SHARE Algorithm is similar). On the other hand, we see in Figure 7 that the STATIC-EXPERT Algorithm also "learned" the best expert for segment 1. However, the STATIC-EXPERT Algorithm is unable to shift the relative weight sufficiently quickly, i.e., it takes the length of the second segment to partially "unlearn" the best expert of the first segment. The relative weights of the best experts for segments one and two essentially perform a "random walk" during the third segment. In the final segment, the relative weight of the best expert for segment three also performs a "random walk." In summary, we see these simulations as evidence that the Fixed-share and Variable-share Updates are necessary to track shifting experts.

11. Conclusion

In this paper, we essentially gave a reduction for any multiplicative update algorithm that works well compared to the best expert for arbitrary segments of examples, to an algorithm that works well compared to the best partition, i.e., a concatenation of segments. Two types of share updates were analyzed. The FIXED-SHARE Algorithm works well when the loss function can be unbounded, and the VARIABLE-SHARE Algorithm is suitable for the case when the range of the loss lies in [0,1]. The first method is essentially the same as the one used in the WML algorithm of (Littlestone & Warmuth, 1994) and a recent alternate developed in (Auer & Warmuth, 1998) for learning shifting disjunctions. When the loss is the discrete loss (as in classification problems), then these methods are simple and effective if the algorithm only updates after a mistake occurs (i.e., conservative updates). Our second method, the Variable-share Update, is more sophisticated. In particular, if one expert predicts perfectly for a while, then it can collect all the weight. However, if this expert is starting to incur large loss, then it shares weight with the other experts, helping the next best expert to recover its weight from zero.

The methods presented here and in (Littlestone & Warmuth, 1994) have inspired a number of recent papers. Auer & Warmuth (1998) adapted the Winnow algorithm to learn shifting disjunctions. Comparing against the best shifting disjunction is more complicated than comparing against the best expert. However, since this is a classification problem a simple Sharing Update similar to the Fixed-share Update is sufficient. Our focus in this paper was to track the prediction of the best expert for the same class of loss functions for which the original STATIC-EXPERT Algorithm of Vovk was developed (Vovk, 1998; Haussler et al., 1998).

Our share updates have been applied experimentally for predicting disk idle times (Helmbold et al., 1996) and for the on-line management of investment portfolios (Singer, 1997). In addition, a reduction has been shown between expert and metrical task systems algorithms (Blum & Burch, 1997). The Share Update has been used successfully in the new domain of metrical task systems. A natural probabilistic interpretation of the Share algorithms has recently been given in (Vovk, 1997).

In any particular application of the Share algorithms, it is necessary to consider how to choose the parameter α. Theoretical techniques exist for the FIXED-SHARE Algorithm for eliminating the need to choose the value of α ahead of time. One method for tuning parameters (among other things) is the "specialist" framework of (Freund, Schapire, Singer & Warmuth, 1997), even though the bounds produced this way are not always optimal. Another method incorporates a prior distribution on all possible values of α. For the sake of simplicity we have not discussed these methods (Herbster, 1997; Vovk, 1997; Singer, 1997) in this paper.

Acknowledgments

We would like to thank Peter Auer, Phillip Long, Robert Schapire, and Volodya Vovk for valuable discussions. We also thank the anonymous referees for their helpful comments. The support of the NSF grant CCR-9700201 is acknowledged.

Notes

1. The discrete loss is defined to be
$$L_{\text{dis}}(p,q) = \begin{cases} 0 & p = q \\ 1 & p \neq q. \end{cases}$$

2. Note that $D(p\|q) = L_{ent}(p,q)$. We use the $D(p\|q)$ notation here as is customary in information theory.
3. If we replace the assumption that $k \geq \hat{k}$ by $2\hat{k} \leq \hat{\ell}$, we obtain a bound where the final term $c\hat{k}$ is replaced by $2c\hat{k}$.
4. Vovk has recently proved a sharper bound for this algorithm (Vovk, 1997):
$$L(S,A) \leq c\ln n + c[\eta + \ln\frac{1}{1-\alpha}]L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S)) + ck[\eta + \ln\frac{1-e^{-\eta}+\alpha e^{-\eta}}{\alpha} + \ln(n-1)].$$

5. Unlike Corollary 2 we do not need a lower bound on k.
6. We call the partition described by the segment boundaries 1, 201, 401, and 601, the best partition with respect to the tradeoff between k and $L(\mathcal{P}_{\ell,n,k,\mathbf{t},\mathbf{e}}(S))$, as expressed implicitly in Theorem 2.

References

Auer, P. & Warmuth, M. K. (1998). Tracking the best disjunction. *Machine Learning*, this issue.

Blum, A. & Burch, C. (1997). On-line learning and the metrical task system. In *Proceedings of the 10th Annual Workshop on Computational Learning Theory*. ACM Press, New York, NY.

Cesa-Bianchi, N., Freund, Y., Haussler, D., Helmbold, D. P., Schapire, R. E., & Warmuth, M. K. (1997). How to use expert advice. *Journal of the ACM, 44(3)*, 427–485.

Cover, T. & Thomas, J. (1991). *Elements of Information Theory*. Wiley.

Feder, M., Merhav, N., & Gutman, M. (1992). Universal prediction of individual sequences. *IEEE Transactions on Information Theory, 38*, 1258–1270.

Freund, Y. & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences ,55(1)*, 119–139.

Freund, Y., Schapire, R. E., Singer, Y., & Warmuth, M. K. (1997). Using and combining predictors that specialize. In *Proceedings of the Twentyninth Annual ACM Symposium on Theory of Computing*.

Haussler, D., Kivinen, J., & Warmuth, M. K. (1998). Sequential prediction of individual sequences under general loss functions. *IEEE Transactions on Information Theory*. To appear.

Helmbold, D. P., Kivinen, J., & Warmuth, M. K. (1995). Worst-case loss bounds for sigmoided linear neurons. In *Proceedings of the 1995 Neural Information Processing Conference*, (pp. 309–315). MIT Press, Cambridge, MA.

Helmbold, D.P., Long, D.D.E., & Sherrod, B. (1996). A dynamic disk spin-down technique for mobile computing. In *Proceedings of the Second Annual ACM International Conference on Mobile Computing and Networking*. ACM/IEEE.

Herbster, M. (1997). Tracking the best expert II. Unpublished Manuscript.

Herbster, M. & Warmuth, M. K. (1995). Tracking the best expert. In *Proceedings of the 12th International Conference on Machine Learning*, (pp. 286–294). Morgan Kaufmann.

Kivinen, J. & Warmuth, M. K. (1997). Additive versus exponentiated gradient updates for linear prediction. *Information and Computation, 132(1)*, 1–64.

Littlestone, N. (1988). Learning when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning, 2*, 285–318.

Littlestone, N. (1989). *Mistake Bounds and Logarithmic Linear-threshold Learning Algorithms*. PhD thesis, Technical Report UCSC-CRL-89-11, University of California Santa Cruz.

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

Singer, Y. (1997). Towards realistic and competitive portfolio selection algorithms. Unpublished Manuscript.

Vovk, V. (1998). A game of prediction with expert advice. *Journal of Computer and System Sciences*. To appear.
Vovk, V. (1997). Derandomizing stochastic prediction strategies. In *Proceedings of the 10th Annual Workshop on Computational Learning Theory*. ACM Press, New York, NY.
Warmuth, M. K. (1997). Predicting with the dot-product in the experts framework. Unpublished Manuscript.

Received September 22, 1997
Accepted December 12, 1997
Final Manuscript February 7, 1998