Mach Learn (2008) 72: 21–37
DOI 10.1007/s10994-008-5060-z

Regret to the best vs. regret to the average

Eyal Even-Dar · Michael Kearns · Yishay Mansour · Jennifer Wortman

Received: 30 August 2007 / Revised: 28 March 2008 / Accepted: 4 April 2008 /
Published online: 13 May 2008

Abstract We study online regret minimization algorithms in an experts setting. In this setting, the algorithm chooses a distribution over experts at each time step and receives a gain that is a weighted average of the experts' instantaneous gains. We consider a bicriteria setting, examining not only the standard notion of regret to the best expert, but also the regret to the average of all experts, the regret to any given fixed mixture of experts, or the regret to the worst expert. This study leads both to new understanding of the limitations of existing no-regret algorithms, and to new algorithms with novel performance guarantees. More specifically, we show that *any* algorithm that achieves only $O(\sqrt{T})$ cumulative regret to the best expert on a sequence of T trials must, in the worst case, suffer regret $\Omega(\sqrt{T})$ to the average, and that for a wide class of update rules that includes many existing no-regret algorithms (such as Exponential Weights and Follow the Perturbed Leader), the product of the regret to the best and the regret to the average is, in the worst case, $\Omega(T)$. We then describe and analyze two alternate new algorithms that both achieve cumulative regret only

Editors: Claudio Gentile, Nader H. Bshouty.

A preliminary version of this work appeared in the *Proceedings of the Twentieth Annual Conference on Learning Theory*, 2007 (Even-Dar et al. 2007).

This work was done while E. Even-Dar was a post doctoral researcher at the University of Pennsylvania.

Y. Mansour was supported in part by grant no. 1079/04 from the Israel Science Foundation, a grant from BSF, an IBM faculty award, and the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This paper reflects only the authors' views.

E. Even-Dar · Y. Mansour
Google Research, 76 Ninth Avenue, New York, NY 10011, USA

M. Kearns · J. Wortman (✉)
Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104, USA
e-mail: wortmanj@seas.upenn.edu

Y. Mansour
Department of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel

$O(\sqrt{T}\log T)$ to the best expert and have only *constant* regret to any given fixed distribution over experts (that is, with no dependence on either T or the number of experts N). The key to the first algorithm is the gradual increase in the "aggressiveness" of updates in response to observed divergences in expert performances. The second algorithm is a simple twist on standard exponential-update algorithms.

Keywords Online learning · Regret minimization · Prediction with expert advice · Lower bounds

1 Introduction

Beginning at least as early as the 1950s, the long and still-growing literature on no-regret learning has established the following type of result. On any sequence of T trials in which the predictions of N "experts" are observed, it is possible to maintain a dynamically weighted prediction whose cumulative regret to the best single expert *in hindsight* (that is, after the full sequence has been revealed) is $O(\sqrt{T\log N})$, with absolutely no statistical assumptions on the sequence. Such results are especially interesting in light of the fact that even in known *stochastic* models, there is a matching lower bound of $\Omega(\sqrt{T\log N})$. The term "no-regret" derives from the fact that the per-step regret is only $O(\sqrt{(\log N)/T})$, which approaches zero as T becomes large.

In this paper we revisit no-regret learning, but with a bicriteria performance measure that is of both practical and philosophical interest. More specifically, in addition to looking at the cumulative regret to the *best* expert in hindsight, we simultaneously analyze the regret to the *average* gain of all experts (or more generally, any fixed weighting of the experts). For comparisons to the average, the gold standard will be only *constant* regret (independent of T and N). Note that considering regret to the average in isolation, *zero* regret is easily achieved by simply leaving the weights uniform at all times.

We consider a setting in which each expert receives a bounded (e.g., in $[0, 1]$) gain at each time step. The gain of the algorithm on a given time step is then a weighted average of these expert gains; this can be interpreted as the expected gain the algorithm would receive if it chose a single expert to follow on each time step according to its current distribution. The regret of the algorithm is measured in terms of cumulative expected gains over time. Our results establish strict trade-offs between regret to the best expert and the regret to the average in this setting, demonstrate that most known algorithms manage this trade-off poorly, and provide new algorithms with near optimal bicriteria performance. On the practical side, our new algorithms augment traditional no-regret results with a "safety net": while still managing to track the best expert near-optimally, they are guaranteed to never underperform the average (or any other given fixed weighting of experts) by more than just constant regret. On the philosophical side, the bicriteria analyses and lower bounds shed new light on prior no-regret algorithms, showing that the unchecked aggressiveness of their updates can indeed cause them to badly underperform simple benchmarks like the average.

Viewed at a suitably high level, many existing no-regret algorithms have a similar flavor. These algorithms maintain a distribution over the experts that is adjusted according to performance. Since we would like to compete with the best expert, a "greedy" or "momentum" algorithm that rapidly adds weight to an outperforming expert (or set of experts) is natural. Most known algorithms shift weight between competing experts at a rate proportional to $1/\sqrt{T}$, in order to balance the tracking of the current best expert with the possibility of this expert's performance suddenly dropping. Updates on the scale of $1/\sqrt{T}$ can be viewed

Table 1 Summary of lower bounds and algorithmic results presented in this paper

Summary of lower bounds		
Algorithm:	*If Regret to Best Is:*	*Then Regret to Average Is:*
Any Algorithm	$O(\sqrt{T})$	$\Omega(\sqrt{T})$
Any Algorithm	$\leq \sqrt{T \log T}/10$	$\Omega(T^{\epsilon})$
Any Difference Algorithm	$O(T^{\frac{1}{2}+\alpha})$	$\Omega(T^{\frac{1}{2}-\alpha})$

Summary of algorithmic results			
Algorithm:	*Regret to Best:*	*Regret to Average:*	*Regret to Worst:*
Phased Aggression	$O(\sqrt{T \log N}(\log T + \log\log N))$	O(1)	O(1)
D-Prod	$O(\sqrt{T \log N} + \sqrt{T/\log N} \log T)$	O(1)	O(1)
BestWorst	$O(N\sqrt{T \log N})$	$O(\sqrt{T \log N})$	0
EW	$O(T^{\frac{1}{2}+\alpha} \log N)$	$O(T^{\frac{1}{2}-\alpha})$	$O(T^{\frac{1}{2}-\alpha})$

as "aggressive", at least in comparison to the minimal average update of $1/T$ required for any interesting learning effects. (If updates are $o(1/T)$, the algorithm cannot make even a constant change to any given weight in T steps.)

How poorly can existing regret minimization algorithms perform with respect to the average? Consider a sequence of gains for two experts where the gains for expert 1 are $1, 0, 1, 0, \ldots$, while the gains for expert 2 are $0, 1, 0, 1, \ldots$. Typical regret minimization algorithms (such as Exponential Weights, Littlestone and Warmuth 1994; Freund 2003, Follow the Perturbed Leader, Kalai and Vempala 2005, and the Prod algorithm, Cesa-Bianchi et al. 2007) will yield a gain of $T/2 - \Theta(\sqrt{T})$, meeting their guarantee of $O(\sqrt{T})$ regret with respect to the best expert. However, this performance leaves something to be desired. Note that in this example the performance of the best expert, worst expert, and average of the experts is identically $T/2$. Thus all of the algorithms mentioned above actually suffer a regret to the average (and to the worst expert) of $\Omega(\sqrt{T})$. The problem stems from the fact that in all even time steps the probability of expert 1 is exactly $1/2$. After expert 1 observes a gain of 1 we increase its probability by $c/\sqrt{T}$, where the precise value of c depends on the specific algorithm. Therefore in odd steps the probability of expert 2 is only $(1/2 - c/\sqrt{T})$. Note that adding a third expert, which is defined as the average of the original two, would not change this.[1]

This paper establishes a sequence of results that demonstrate the inherent tension between regret to the best expert and the average, illuminates the problems of existing algorithms in managing this tension, and provides new algorithms that enjoy optimal bicriteria performance guarantees.

On the negative side, we show that *any* algorithm that has a regret of $O(\sqrt{T})$ to the best expert must suffer a regret of $\Omega(\sqrt{T})$ to the average in the worst case. We also show that any regret minimization algorithm that achieves at most $\sqrt{T \log T}/10$ regret to the best expert, must, in the worst case, suffer regret $\Omega(T^{\epsilon})$ to the average, for some constant $\epsilon \geq 0.02$. These lower bounds are established even when $N = 2$.

[1] The third expert would clearly have a gain of $1/2$ at every time step. At odd time steps, the weight of the first expert would be $1/3 + c/\sqrt{T}$, while that of the second expert would be $1/3 - c/\sqrt{T}$, resulting in a regret of $\Omega(\sqrt{T})$ to the average.

On the positive side, we describe a new algorithm, *Phased Aggression*, that almost matches the lower bounds above. Given any algorithm whose cumulative regret to the best expert is at most R (which may be a function of T and N but not of any data-dependent measures), we can use it to derive an algorithm whose regret to the best expert is $O(R \log R)$ with only constant regret to the average (or any given fixed distribution over the experts). Using an $O(\sqrt{T \log N})$ regret algorithm, this gives regret to the best of $O(\sqrt{T \log N}(\log T + \log\log N))$. In addition, we show how to use an R-regret algorithm to derive an algorithm with regret $O(NR)$ to the best expert and *zero* regret to the worst expert. These algorithms treat the given R-regret algorithm as a black box.

Phased Aggression is somewhat different from many of the traditional regret minimization algorithms, especially in its use of *restarts* that are driven by observed differences in expert performance. (Restarts have been used previously in the literature, but for other purposes (Cesa-Bianchi and Lugosi 2006).) We show that this difference is no coincidence. For a wide class of update rules that includes many existing algorithms (such as Weighted Majority/Exponential Weights, Follow the Perturbed Leader, and Prod), we show that the product of the regret to the best and the regret to the average is $\Omega(T)$. This establishes a frontier from which such algorithms inherently cannot escape. Furthermore, any point on this frontier can in fact be achieved by such an algorithm (i.e., a standard multiplicative update rule with an appropriately tuned learning rate). However, we show it is possible to circumvent the lower bound by using an algorithm "similar to" Prod to achieve guarantees close to those achieved by *Phased Aggression* without the use of restarts. This algorithm, *D-Prod*, escapes the lower bound by using a modified update rule that directly depends on the average of the experts' instantaneous gains at each time step.

It is worth noting that it is not possible in general to guarantee $o(\sqrt{T})$ regret to any arbitrary *pair* of distributions, D_1 and D_2. Consider a setting in which there are only two experts. Suppose distribution D_1 places all weight on one expert, while distribution D_2 places all weight on a second. Competing simultaneously with both distributions is then equivalent to competing with the best expert, so we cannot expect to do better than known lower bounds of $\Omega(\sqrt{T})$.

Related work Previous work by Auer et al. (2002) considered adapting the learning rate of expert algorithms gradually. However, the goal of their work was to get an any-time regret bound without using the standard doubling technique and thus it is not surprising that their algorithm performance under the bicriteria setting is similar to the other existing algorithms. Vovk (1998) also considered trade-offs in best expert algorithms. His work examined for which values of a and b it is possible for an algorithm's gain to be bounded by $aG_{best,T} + b \log N$, where $G_{best,T}$ is the gain of the best expert.

2 Preliminaries

We consider the classic experts framework, in which each expert $i \in \{1, \ldots, N\}$ receives a gain $g_{i,t} \in [0, 1]$ at each time step t.[2] The cumulative gain of expert i up to time t is $G_{i,t} = \sum_{t'=1}^{t} g_{i,t'}$. We denote the average cumulative gain of the experts by time t as $G_{avg,t} = (1/N) \sum_{i=1}^{N} G_{i,t}$, and the gain of the best and worst expert as $G_{best,t} = \max_i G_{i,t}$ and $G_{worst,t} = \min_i G_{i,t}$. For any fixed distribution D over the experts, we define the gain of this distribution to be $G_{D,t} = \sum_{i=1}^{N} D(i) G_{i,t}$.

[2] All results presented in this paper can be generalized to hold for instantaneous gains in any bounded region.

At each time t, an algorithm A assigns a weight $w_{i,t}$ to each expert i. These weights are normalized to probabilities $p_{i,t} = w_{i,t}/W_t$ where $W_t = \sum_i w_{i,t}$. Algorithm A then receives a gain $g_{A,t} = \sum_{i=1}^{N} p_{i,t} g_{i,t}$. The cumulative gain of algorithm A up to time t is $G_{A,t} = \sum_{t'=1}^{t} g_{A,t'} = \sum_{t'=1}^{t} \sum_{i=1}^{N} p_{i,t'} g_{i,t'}$.

The standard goal of an algorithm in this setting is to minimize the regret to the best expert at a fixed time T. In particular, we would like to minimize the regret $R_{best,A,T} = \max\{G_{best,T} - G_{A,T}, 1\}$.[3] In this work, we are simultaneously concerned with minimizing both this regret and the regret to the average and worst expert, $R_{avg,A,T} = \max\{G_{avg,T} - G_{A,T}, 1\}$ and $R_{worst,A,T} = \max\{G_{worst,T} - G_{A,T}, 1\}$ respectively, in addition to the regret $R_{D,A,T}$ to a given distribution D, which is defined similarly.

While in general the bounds on the regret of the algorithms can be defined using the time (for example, a regret of $O(\sqrt{T})$) and these bounds are tight, this is considered a crude estimate and better measures are at hand. We present our results in terms of the maximal absolute gains $G_{max} = \max_i G_{i,T}$.

3 The $\Theta(T)$ frontier for difference algorithms

We begin our results with an analysis of the trade-off between regret to the best and average for a wide class of existing algorithms, showing that the product between the two regrets for this class is $\Theta(T)$. A more general lower bound that holds for *any* algorithm is provided in Sect. 5.

Definition 1 (Difference algorithm) We call an algorithm A a *difference algorithm* if, when $N = 2$ and instantaneous gains are restricted to $\{0, 1\}$, the normalized weights A places on each of the two experts depend only on the difference between the experts' cumulative gains. In other words, A is a difference algorithm if there exists a function f such that when $N = 2$ and $g_{i,t} \in \{0, 1\}$ for all i and t, $p_{1,t} = f(d_t)$ and $p_{2,t} = 1 - f(d_t)$ where $d_t = G_{1,t} - G_{2,t}$.

Exponential Weights (Littlestone and Warmuth 1994; Freund 2003), Follow the Perturbed Leader (Kalai and Vempala 2005), and the Prod algorithm (Cesa-Bianchi et al. 2007) are all examples of difference algorithms; for Prod, this follows from the restriction on the instantaneous gains to $\{0, 1\}$. While a more general definition of the class of difference algorithms might be possible, this simple definition is sufficient to show the lower bound.

3.1 Difference frontier lower bound

Theorem 1 *Let A be any difference algorithm. Then*

$$R_{best,A,T} \cdot R_{avg,A,T} \geq R_{best,A,T} \cdot R_{worst,A,T} = \Omega(T).$$

Proof For simplicity, assume that T is an even integer. We will consider the behavior of the difference algorithm A on two sequences of expert payoffs. Both sequences involve only two experts with instantaneous gains in $\{0, 1\}$. (Since the theorem provides a lower bound, it is sufficient to consider an example in this restricted setting.) Assume without loss of generality that initially $p_{1,1} \leq 1/2$.

[3] This minimal value of 1 makes the presentation of the trade-off "nicer" (for example in the statement of Theorem 1), but has no real significance otherwise.

In the first sequence, S_1, Expert 1 has a gain of 1 at every time step while Expert 2 always has a gain 0. Let ρ be the first time t at which A has $p_{1,t} \geq 2/3$. If such a ρ does not exist, then $R_{best,A,T} = \Omega(T)$ and we are done. Assuming such a ρ does exist, A must have regret $R_{best,A,T} \geq \rho/3$ since it loses at least $1/3$ to the best expert on each of the first ρ time steps and cannot compensate for this later.

Since the probability of Expert 1 increases from $p_{1,1} \leq 1/2$ to at least $2/3$ in ρ time steps in S_1, there must be one time step $\tau \in [2, \rho]$ in which the probability of Expert 1 increased by at least $1/(6\rho)$, i.e., $p_{1,\tau} - p_{1,\tau-1} \geq 1/(6\rho)$. The second sequence S_2 we consider is as follows. For the first τ time steps, Expert 1 will have a gain of 1 (as in S_1). For the last τ time steps, Expert 1 will have a gain of 0. For the remaining $T - 2\tau$ time steps (in the range $[\tau, T - \tau]$), the gain of Expert 1 will alternate $0, 1, 0, 1, \ldots$. Throughout the sequence, Expert 2 will have a gain of 1 whenever Expert 1 has a gain of 0 and a gain of 0 every time Expert 1 has a gain of 1. This implies that each expert has a gain of exactly $T/2$ (and hence $G_{best,T} = G_{avg,T} = G_{worst,T} = T/2$).

During the period $[\tau, T - \tau]$, consider a pair of consecutive times such that $g_{1,t} = 0$ and $g_{1,t+1} = 1$. Since A is a difference algorithm we have that $p_{1,t} = p_{1,\tau}$ and $p_{1,t+1} = p_{1,\tau-1}$. The gain of algorithm A in time steps t and $t + 1$ is $(1 - p_{1,\tau}) + p_{1,\tau-1} \leq 1 - 1/(6\rho)$, since $p_{1,\tau} - p_{1,\tau-1} \geq 1/(6\rho)$. In every pair of time steps t and $T - t$, for $t \leq \tau$, the gain of A in those times steps is exactly 1, since the difference between the experts is identical at times t and $T - t$, and hence the probabilities are identical. This implies that the total gain of the algorithm A is at most

$$\tau + \frac{T - 2\tau}{2}\left(1 - \frac{1}{6\rho}\right) \leq \frac{T}{2} - \frac{T - 2\tau}{12\rho}.$$

On sequence S_1, the regret of algorithm A with respect to the best expert is $\Omega(\rho)$. Therefore, if $\rho \geq T/4$ we are done. Otherwise, on sequence S_2, the regret with respect to the average and worst is $\Omega(T/\rho)$. The theorem follows. □

3.2 A difference algorithm achieving the frontier

We now show that the standard Exponential Weights (EW) algorithm with an appropriate choice of the learning rate parameter η (Freund 2003) is a difference algorithm achieving the trade-off described in Sect. 3.1, thus rendering it tight for this class. Recall that for all experts i, EW assigns initial weights $w_{i,1} = 1$, and at each subsequent time t, updates weights with $w_{i,t+1} = e^{\eta G_{i,t}} = w_{i,t} e^{\eta g_{i,t}}$. The probability with which expert i is chosen at time t is then given by $p_{i,t} = w_{i,t}/W_t$ where $W_t = \sum_{j=1}^{N} w_{j,t}$.

Theorem 2 *Let $G^* \leq T$ be an upper bound on G_{max}. For any α such that $0 \leq \alpha \leq 1/2$, let $EW = EW(\eta)$ with $\eta = (G^*)^{-(1/2+\alpha)}$. Then*

$$R_{best,EW,T} \leq (G^*)^{1/2+\alpha}(1 + \ln N)$$

and

$$R_{avg,EW,T} \leq (G^*)^{1/2-\alpha}.$$

Proof These bounds can be derived using a series of bounds on the quantity $\ln(W_{T+1}/W_1)$. First we bound this quantity in terms of the gain of the best expert and the gain of EW. This piece of the analysis is standard (see, for example, Theorem 2.4 in Cesa-Bianchi and

Lugosi 2006). The first piece of the bound follows from the fact that $W_1 = N$ and that $w_{best,t+1} \leq W_{t+1}$.

$$\eta G_{best,T} - \ln N \leq \ln\left(\frac{W_{T+1}}{W_1}\right).$$

The second pieces follows from a simple application of Taylor approximation.

$$\ln\left(\frac{W_{T+1}}{W_1}\right) \leq \left(\eta + \eta^2\right) G_{EW,T}. \tag{1}$$

Therefore, we derive that

$$G_{best,T} - G_{EW,T} \leq \eta G_{EW,T} + \frac{\ln N}{\eta}.$$

Next we bound the same quantity in terms of the average cumulative gain, using the fact that the arithmetic mean of a set of nonnegative numbers is always greater than or equal to the geometric mean.

$$\begin{aligned}\ln\left(\frac{W_{T+1}}{W_1}\right) &= \ln\left(\frac{\sum_{i=1}^{N} w_{i,T+1}}{N}\right) \geq \ln\left(\left(\prod_{i=1}^{N} w_{i,T+1}\right)^{\frac{1}{N}}\right) \\ &= \frac{1}{N}\sum_{i=1}^{N} \ln w_{i,T+1} = \frac{1}{N}\sum_{i=1}^{N} \eta G_{i,T} = \eta G_{avg,T}. \end{aligned} \tag{2}$$

Combined with the upper bound in (1), this gives us

$$G_{avg,T} - G_{EW,T} \leq \eta G_{EW,T}.$$

Note that if $G_{best,T} \leq G_{EW,T}$, both the regret to the best expert and the regret to the average will be minimal, so we can assume this is not the case and replace the term $G_{EW,T}$ on the right hand side of these bounds with $G_{best,T}$ which is in turn bounded by G^*. This yields the following pair of bounds.

$$G_{best,T} - G_{EW,T} \leq \eta G^* + \ln N/\eta$$
$$G_{avg,T} - G_{EW,T} \leq \eta G^*.$$

By changing the value of η, we can construct different trade-offs between the two bounds. Setting $\eta = (G^*)^{-(1/2+\alpha)}$ yields the desired result. □

This trade-off can be generalized to hold when we would like to compete with an arbitrary distribution D by initializing $w_{i,1} = D(i)$ and substituting an alternate inequality into (2). The $\ln(N)$ term in the regret to the best expert will be replaced by $max_{i \in N} \ln(1/D(i))$, making this practical only for distributions that lie inside the probability simplex and not too close to the boundaries.

4 Breaking the difference frontier

The results so far have established a $\Theta(T)$ frontier on the product of regrets to the best and average experts for difference algorithms. In this section, we will show how this frontier can

BestWorst (A, R)**, where** A **guarantees** $R_{best,A,T} \leq R$
while $(G_{best,t} - G_{worst,t} \leq NR)$ **do**
Use probabilities $p_{i,t} = \frac{1}{N}$ for all i
end
Reset and run algorithm A for all remaining time steps.

Fig. 1 BestWorst algorithm for N experts

be broken by non-difference algorithms that gradually increase the aggressiveness of their updates via a series of restarts invoked by observed differences in performance so far. As a warm-up, we first show how a very simple algorithm that is not a difference algorithm can enjoy standard regret bounds compared to the best expert in terms of T (though worse in terms of N), while having *zero* cumulative regret to the worst. In Sects. 4.2 and 4.3, we present two alternative algorithms that compete well with both the best expert and the average with only logarithmic dependence on N.

4.1 Regret to the best and worst experts

Using a standard regret-minimization algorithm as a black box, we can produce a very simple algorithm, *BestWorst*, that achieves a clear trade-off between regret to the best expert and regret to the worst expert. Let A be a regret minimization algorithm such that $R_{best,A,T} \leq R$ for some R which may be a function of T and N but not of any data dependent measures. We define the modified algorithm *BestWorst(A)* as follows. While the difference between the cumulative gains of the best and worst experts is smaller than NR, *BestWorst(A)* places equal weight on each expert, playing the average. After the first time τ at which this condition is violated, it begins running a fresh instance of algorithm A and continues to use A until the end of the sequence.

Until time τ, this algorithm must be performing at least as well as the worst expert since it is playing the average. At time τ, the algorithm's gain must be R more than that of the worst expert since the gain of the best expert is NR above the gain of the worst. Now since from time τ algorithm A is run, we know that the gain of *BestWorst(A)* in the final $T - \tau$ time steps will be within R of the gain of the best expert. Therefore, *BestWorst(A)* will maintain a lead over the worst expert. In addition, the regret of the algorithm to the best expert will be bounded by NR, since up to time τ it will have a regret of at most $(N-1)R$ with respect to the best expert. This establishes the following theorem.

Theorem 3 *Let A be a regret minimization algorithm with regret at most R to the best expert and let BW be* BestWorst(A,R). *Then*

$$R_{best,BW,T} \leq NR$$

and

$$G_{BW,T} \geq G_{worst,T}.$$

It follows immediately that using a standard regret minimization algorithm with $R = O(\sqrt{T \log N})$ as the black box, we can achieve a regret of $O(N\sqrt{T \log N})$ to the best expert while maintaining a lead over the worst.

PhasedAggression (A, R, D)
for $k = 1\ to\ \lfloor \log(R) \rfloor$ **do**
  Let $\eta = 2^{k-1}/R$
  Reset and run a new instance of A
  while $(G^p_{best,t} - G^p_{D,t} < 2R)$ **do**
    Feed A with the previous gains g_{t-1} and let q_t be its distribution
    Use $p_t = \eta q_t + (1 - \eta)D$
  end
end
Reset and run a new instance of A until time T

Fig. 2 The *Phased Aggression* algorithm for N experts

4.2 Phased aggression

Again using any standard regret-minimization algorithm as a black box, we can produce an algorithm, *Phased Aggression*, that achieves a trade-off between regret to the best expert and regret to the average without sacrificing too much in terms of the dependence on N. Figure 2 shows *Phased Aggression*. This algorithm can achieve a constant regret to *any* specified distribution D, not only the average, with no change to the bounds. The name of the algorithm refers to the fact that it operates in distinct phases separated by restarts, with each phase more aggressive than the last.

The idea behind the algorithm is rather simple. We take a regret minimization algorithm A, and mix between A and the target distribution D. As the gain of the best expert exceeds the gain of D by larger amounts, we put more and more weight on the regret minimization algorithm A, "resetting" A to its initial state at the start of each phase. Once the weight on A has been increased, it is never decreased again. In other words, in each successive phase of this algorithm (or reduction), weight is moved from something that is not learning at all (the fixed distribution D) to an algorithm that is implicitly learning aggressively (the given algorithm A). New phases are invoked only in response to greater and greater outperformance by the current best expert, allowing the amount of aggression to increase only as needed.

Theorem 4 *Let A be any algorithm with regret R to the best expert, D be any distribution, and PA be an instantiation of* PhasedAggression(A, R, D). *Then*

$$R_{best,PA,T} \leq 2R(\log R + 1)$$

and

$$R_{D,PA,T} \leq 1.$$

Proof We will again analyze the performance of the algorithm compared to the best expert and the distribution D both during and at the end of any phase k. First consider any time t during phase k. The regret of the algorithm is split between the regret of the fixed mixture and the regret of the no-regret algorithm according to their weights. Since A is an R-regret algorithm its regret to both the best expert and to the distribution D is bounded by R, and thus the regret of the algorithm due to the weight on A is $2^{k-1}/R$ times R. With the remaining $1 - (2^{k-1}/R)$ weight, the regret to the best expert is bounded by $2R$ since

$G^p_{best,t} - G^p_{D,t} < 2R$ during the phase, and its regret to distribution D is 0. Thus at any time t during phase k we have

$$G^p_{best,t} - G^p_{PA,t} \leq R\left(\frac{2^{k-1}}{R}\right) + 2R\left(1 - \frac{2^{k-1}}{R}\right) \leq 2R$$

and

$$G^p_{D,t} - G^p_{PA,t} \leq R\left(\frac{2^{k-1}}{R}\right) = 2^{k-1}.$$

Now consider what happens when the algorithm exits phase k. A phase is only exited at some time t such that $G^p_{best,t} - G^p_{D,t} \geq 2R$. Since A is R-regret, its gain (in the current phase) will be within R of the gain of the best expert, resulting in the algorithm PA gaining a *lead* over distribution D for the phase: $G^p_{PA,t} - G^p_{D,t} \geq R(2^{k-1}/R) = 2^{k-1}$.

Combining these inequalities, it is clear that if the algorithm ends in phase k at time T, then

$$G_{best,T} - G_{PA,T} \leq 2Rk \leq 2R(\log R + 1)$$

and

$$G_{D,T} - G_{PA,T} \leq 2^{k-1} - \sum_{j=1}^{k-1} 2^{j-1} = 2^{k-1} - (2^{k-1} - 1) = 1.$$

These inequalities hold even when the algorithm reaches the final phase and has all of its weight on A, thus proving the theorem. □

4.3 D-Prod

While Exponential Weights and Prod are both difference algorithms and cannot avoid the $\Omega(T)$ frontier lower bound, it is possible to create an algorithm with exponential updates that can achieve guarantees similar to those of *Phased Aggression* without requiring the use of restarts. This can be accomplished via a simple algorithm that we refer to as *D-Prod* since its update rules and analysis are inspired by those of Prod.

D-Prod differs from Prod in two important ways that together allow it to compete well with the best expert while simultaneously guaranteeing only constant regret to a given fixed distribution D (Zhang 2007). First, an additional expert (denoted expert 0) representing the distribution D is added with a large prior weight. Second, the update rule is modified to take into account the *difference* in performance between each expert and the distribution D rather than the gains of each expert alone. Because of this second modification, *D-Prod* is *not* a difference algorithm and is able to avoid the $\Omega(T)$ frontier.

Formally, let $g_{i,t} \in [0, 1]$ be the gain of expert i at time t as before for $i \in \{1, \ldots, N\}$, and let $g_{0,t}$ be the instantaneous gain of the distribution D (or the special expert 0). Each expert i starts with an initial or prior weight $w_{i,1} = \mu_i$, and updates are made as follows:

$$w_{i,t+1} = w_{i,t}(1 + \eta(g_{i,t} - g_{0,t})).$$

Lemma 1 *For any expert i (including the special expert 0), for any $\eta \leq 1/2$,*

$$G_{D-Prod,T} \geq G_{i,T} + \frac{\ln(\mu_i)}{\eta} - \eta \sum_{t=1}^{T} (g_{i,t} - g_{0,t})^2.$$

The proof is nearly identical to the proof of Lemma 2 of Cesa-Bianchi et al. (2007). Notice that when $i = 0$ (the special expert), the last term in the bound is 0. The following theorem shows how to set the parameters η and μ to achieve a constant error rate to the distribution D without losing much with respect to the best expert.

Theorem 5 *Let* $\eta = \sqrt{\ln N/T}$, $\mu_0 = 1 - \eta$, *and* $\mu_i = \eta/N$ *for* $i \in \{1, \ldots, N\}$. *Then*

$$R_{best,D-Prod,T} = O\left(\sqrt{T \ln N} + \sqrt{\frac{T}{\ln N}} \ln T\right)$$

and

$$R_{D,D-Prod,T} = O(1).$$

5 A general lower bound

So far we have seen that a wide class of existing algorithms (namely all difference algorithms) is burdened with a stark best/average regret trade-off, but that this frontier can be avoided by simple algorithms that tune how aggressively they update, in phases modulated by the observed payoffs so far. What is the limit of what can be achieved in our bicriteria regret setting?

In this section we show a pair of general lower bounds that hold for *all* algorithms. The bounds are stated and proved for the average but once again hold for any fixed distribution D. These lower bounds come close to the upper bound achieved by the algorithms described in the previous section.

Theorem 6 *Any algorithm with regret* $O(\sqrt{T})$ *to the best expert must have regret* $\Omega(\sqrt{T})$ *to the average. Furthermore, any algorithm with regret at most* $\sqrt{T \log T}/10$ *to the best expert must have regret* $\Omega(T^\epsilon)$ *to the average for some positive constant* $\epsilon \geq 0.02$.

More specifically, we show that for any constant $\alpha > 0$, there exists a constant $\beta > 0$ such that for sufficiently large values of T (i.e. $T > (150\alpha)^2$), for any algorithm A, there exists a sequence of gains $\mathbf{g}$ of length T such that if $R_{best,A,T} \leq \alpha\sqrt{T}$ then $R_{avg,A,T} \geq \beta\sqrt{T}$ even when $N = 2$. Additionally, for any constant $\alpha' \leq 1/10$ there exist constants $\beta' > 0$ and $\epsilon > 0$ such that for sufficiently large values of T (i.e., $T > 2^{(10\alpha')^2}$), for any algorithm A, there exists a sequence of gains of length T such that if $R_{best,A,T} \leq \alpha'\sqrt{T \log T}$ then $R_{avg,A,T} \geq \beta' T^\epsilon$.

The proof of this theorem begins by defining a procedure for creating a "bad" sequence $\mathbf{g}$ of expert gains for a given algorithm A. This sequence can be divided into a number of (possibly noncontiguous) segments. By first analyzing the maximum amount that the algorithm can gain over the average and the minimum amount it can lose to the average in each segment, and then bounding the total number of segments possible under the assumption that an algorithm is no-regret, we can show that it is not possible for an algorithm to have $O(\sqrt{T})$ regret to the best expert without having $\Omega(\sqrt{T})$ regret to the average. The remainder of the section is devoted to the proof of Theorem 6.

Fix a constant $\alpha > 0$. Fig. 3 shows a procedure that, given an algorithm A, generates a sequence of expert gains $\mathbf{g}$ of length $T > (150\alpha)^2$ such that $\mathbf{g}$ will be "bad" for A. In this procedure, the variable d_t keeps track of the difference between the gains of the two experts at time t. At each time step, this difference either increases by one or decreases

GenerateBadSeq(A, f, γ)
$t = 1$; $G_{avg,0} = G_{A,0} = d_0 = 0$;
while $(G_{avg,t-1} - G_{A,t-1} \leq 0.115\sqrt{T}/\gamma)$ **do**
  $p_{1,t} = A(\mathbf{g})$; $p_{2,t} = 1 - A(\mathbf{g})$
  if $(d_{t-1} = 0)$ **then**
    if $\left(p_{1,t} \leq \frac{1}{2}\right)$ **then**
      $g_{1,t} = 1$; $g_{2,t} = 0$; $last(|d_{t-1}|) = p_{1,t}$;
    else
      $g_{1,t} = 0$; $g_{2,t} = 1$; $last(|d_{t-1}|) = p_{2,t}$;
    end
  else
    $i_t = \text{argmax}_i\, G_{i,t-1}$; $j_t = \text{argmin}_j\, G_{j,t-1}$;
    $last(|d_{t-1}|) = p_{i_t,t}$;
    $\epsilon_t = p_{i_t,t} - last(|d_{t-1} - 1|)$;
    if $(\epsilon_t \leq f(|d_{t-1}|))$ **then**
      $g_{i_t,t} = 1$; $g_{j_t,t} = 0$;
    else
      $g_{i_t,t} = 0$; $g_{j_t,t} = 1$;
    end
  end
  $G_{A,t} = G_{A,t-1} + p_{1,t}g_{1,t} + p_{2,t}g_{2,t}$;
  $G_{avg,t} = G_{avg,t-1} + (g_{1,t} + g_{2,t})/2$;
  $d_t = d_{t-1} + g_{1,t} - g_{2,t}$;
  $t = t + 1$;
end
$g_{1,t} = g_{2,t} = \frac{1}{2}$ for the rest of the sequence

Fig. 3 Algorithm for creating a bad sequence for any algorithm A

by one, since one expert receives a gain of one and the other zero. The variable $last(d)$ holds the probability that the algorithm assigned to the leading expert the most recent time that the distance between expert gains was d. The variable ϵ_t then represents the difference between the probability that the algorithm assigned to the current best expert at the last time step at which the difference in expert gains was smaller than d_{t-1} and the probability that the algorithm assigns to the best expert for the upcoming time step t. This is used by the sequence generation algorithm to ensure that the best expert will only do well when the algorithm does not have "too much" weight on it. The function f and parameter γ used in the procedure will be defined later in the analysis.

The sequence of gains generated by the procedure in Fig. 3 is specifically designed to fool the algorithm A. In particular, whenever the algorithm updates its weights aggressively, the procedure assigns positive gains to the second-place expert, inducing mean reversion. On the contrary, when the algorithm updates its weights conservatively, the procedure assigns positive gains to the best expert, causing added momentum. These competing issues of conservative versus aggressive updates force the algorithm to have "bad" regret to either the best expert or the average on some sequence of gains.

We say that an algorithm A is *f-compliant* (for the specific function f which will be defined shortly) if at every time t we have (1) $\epsilon_t = f(d_{t-1}) \pm \delta$, for an arbitrarily small δ (for example $\delta = 1/T^2$), and (2) $p_{1,t} = p_{2,t} = 1/2$ if $d_{t-1} = 0$. Since δ can be arbitrarily small, we can think of this requirement as enforcing that ϵ_t be *exactly* equal to $f(d_{t-1})$, and

allowing the algorithm to "choose" whether it should be considered larger or smaller. The following lemma implies that given the sequence generation process in Fig. 3, we need only to consider the class of f-compliant algorithms, since for any other algorithm that does not have $\Omega(\sqrt{T})$ regret to the average, there exists an f-compliant algorithm with better gains for which the same sequence is generated.

Lemma 2 *Consider any algorithm A such that for all $t < T$, $(G_{avg,t-1} - G_{A,t-1} \leq 0.115\sqrt{T}/\gamma)$, and let $\mathbf{g} = GenerateBadSeq(A, f, \gamma)$. There exists an f-compliant algorithm A' such that $GenerateBadSeq(A', f, \gamma) = \mathbf{g}$ and at every time $t \leq T$, $g_{A',t} \geq g_{A,t}$.*

Proof First consider any time t at which $d_{t-1} = 0$. When this is the case, the procedure will always assign a gain of 1 to the expert with the lower probability. Thus if A sets $p_{1,t} < p_{2,t}$ or $p_{2,t} < p_{1,t}$, it is possible to achieve a higher gain by setting $p_{1,t} = p_{2,t} = 1/2$ without altering the sequence $\mathbf{g}$ generated by *GenerateBadSeq*.

Suppose $d_{t-1} \neq 0$. We can assume without loss of generality that $d_{t-1} > 0$. Note that when $\epsilon_t \leq f(|d_{t-1}|)$ we have a gain $g_{1,t} = 1$, so maximizing ϵ_t by setting it arbitrarily close to $f(|d_{t-1}|)$ increases the gain without changing $GenerateBadSeq(A', f, \gamma)$. Similarly, when $\epsilon_t > f(|d_{t-1}|)$ we have $g_{2,t} = 1$, so minimizing ϵ_t by setting it arbitrarily close to $f(|d_{t-1}|)$ maximizes the gain of A without changing $GenerateBadSeq(A', f, \gamma)$. In both cases, letting ϵ_t approach $f(|d_{t-1}|)$ is better for the algorithm and thus the modified algorithm A' will always have a higher payoff on $GenerateBadSeq(A', f, \gamma)$. □

Given an f-compliant algorithm, we can write its probabilities as a function of the difference between expert gains. In particular, we define a function $F(d) = 1/2 + \sum_{i=1}^{|d|} f(i)$, where $F(0) = 1/2$. It is easy to verify that an algorithm A that sets the probability of the best expert at time t to $F(d_{t-1})$ is an f-compliant algorithm. Furthermore, as δ approaches 0, *every* f-compliant algorithm will assign expert weights arbitrarily close to these weights. It is convenient to think of the algorithm weights in this way for the next steps of the analysis.

We are now ready to define the function f used in sequence generation. Let

$$f(d) = \frac{2^{m(d)-1}}{\gamma\sqrt{T}} \quad \text{where } m(d) = \left\lceil \frac{16\alpha}{\sqrt{T}}|d| \right\rceil.$$

It then follows that

$$F(d) = \frac{1}{2} + \sum_{i=1}^{|d|} \frac{2^{m(i)-1}}{\gamma\sqrt{T}} \leq \frac{1}{2} + \sum_{j=1}^{m(d)} \frac{2^{j-1}}{\gamma\sqrt{T}} \left(\frac{\sqrt{T}}{16\alpha}\right) \leq \frac{1}{2} + \frac{2^{m(d)}}{16\gamma\alpha}. \tag{3}$$

We next define the (possibly noncontiguous) *m segment* $\mathcal{T}_m$ to be the set of all times t for which $m(d_t) = m$. More explicitly,

$$\mathcal{T}_m = \{t : (m-1)(\sqrt{T}/(16\alpha)) \leq |d_t| < m(\sqrt{T}/(16\alpha))\}.$$

We also need to introduce the notion of *matched times* and *unmatched times*. We define a pair of matched times as two times t_1 and t_2 such that the difference between the cumulative gains the two experts changes from d to $d+1$ by time t_1 and stays at least as high as $d+1$ until changing from $d+1$ back to d at time t_2. More formally, for some difference d, $d_{t_1-1} = d$, $d_{t_1} = d+1$, $d_{t_2} = d$, and for all t such that $t_1 < t < t_2$, $d_t > d$ (which implies that $d_{t_2-1} = d+1$). Clearly each pair of matched times consists of one time step in which

the gain of one expert is 1 and the other 0 while at the other time step the reverse holds. We refer to any time at which one expert has gain 1 while the other has gain 0 that is *not* part of a pair of matched times as an unmatched time. If at any time t we have $d_t = d$, then there must have been d unmatched times at some point before time t. We denote by $\mathcal{M}_m$ and $\mathcal{UM}_m$ the matched and unmatched times in $\mathcal{T}_m$, respectively. These concepts will become important due to the fact that an algorithm will lose with respect to the average for every pair of matched times, but will gain with respect to the average on every unmatched time.

The following lemma quantifies the regret of the algorithm to the best expert and the average of all experts for each pair of matched times.

Lemma 3 *For any f-compliant algorithm A and any pair of matched times t_1 and t_2 in the m segment, the gain of the algorithm from times t_1 and t_2 (i.e., $g_{A,t_1} + g_{A,t_2}$) is $1 - 2^{m-1}/(\gamma\sqrt{T})$, while the gain of the average and the best expert is 1.*

Proof Let $d = d_{t_1} - 1$. Without loss of generality assume that the leading expert is expert 1, i.e., $d \geq 0$. The gain of the algorithm at time t_1 is $p_{1,t_1} = F(d)$, while the gain at t_2 is $p_{2,t_2} = 1 - p_{1,t_2} = 1 - F(d+1) = 1 - (F(d) + f(d))$. Thus the algorithm has a total gain of $1 - f(d) = 1 - 2^{m-1}/(\gamma\sqrt{T})$ for these time steps. □

Our next step is to provide an upper bound for the gain of the algorithm over the average expert from the unmatched times only.

Lemma 4 *The gain of any f-compliant algorithm A in only the unmatched times in the m segment of the algorithm is at most $2^m\sqrt{T}/(256\gamma\alpha^2)$ larger than the gain of the average expert in the unmatched times in segment m, i.e.,*

$$\sum_{t \in \mathcal{UM}_m} \left(g_{A,t} - \frac{1}{2} \right) \leq \frac{2^m \sqrt{T}}{256\gamma\alpha^2}.$$

Proof Since the leading expert does not change in the unmatched times (in retrospect), we can assume w.l.o.g. that it is expert 1. From (3), it follows that

$$\sum_{t \in \mathcal{UM}_m} g_{A,t} - 1/2 \leq \sum_{i=0}^{\frac{\sqrt{T}}{16\alpha}-1} \left(F(d+i) - \frac{1}{2} \right) \leq \frac{2^m}{16\gamma\alpha} \frac{\sqrt{T}}{16\alpha} \leq \frac{2^m\sqrt{T}}{256\gamma\alpha^2}.$$ □

Combining Lemmas 3 and 4, we can compute the number of matched times needed in the m segment in order for the loss of the algorithm to the average from matched times to cancel the gain of the algorithm over the average from unmatched times.

Lemma 5 *For any fixed integer x, if there are at least $T/(128\alpha^2) + x$ pairs of matched times in the m segment, then the gain of any f-compliant algorithm A in the m segment is bounded by the gain of the average expert in the m segment minus $x2^{m-1}/(\gamma\sqrt{T})$, i.e.,*

$$\sum_{t \in \mathcal{T}_m} g_{A,t} \leq \sum_{t \in \mathcal{T}_m} \frac{1}{2} - \frac{2^{m-1}x}{\gamma\sqrt{T}}.$$

Proof From Lemma 4, A cannot gain more than $2^m\sqrt{T}/(256\alpha^2\gamma)$ over the average in the m segment. From Lemma 3, the loss of A with respect to the average for each pair of matched

times is $2^{m-1}/(\gamma\sqrt{T})$. Since there are at least $T/(128\alpha^2)+x$ pairs of matched times, the *total* amount the algorithm loses to the average in the m segment is at least $2^{m-1}x/(\gamma\sqrt{T})$. □

The next lemma bounds the number of segments in the sequence using the fact that A is $\alpha\sqrt{T}$-regret algorithm.

Lemma 6 *For any f-compliant algorithm A such that $R_{best,A,T} < \alpha\sqrt{T}$ and for $\gamma = 2^{48\alpha^2}/\alpha$, there are at most $48\alpha^2$ segments in $\mathbf{g} = GenerateBadSeq(A, f, \gamma)$.*

Proof Once again we assume that leading expert is expert 1. Setting $\gamma = 2^{48\alpha^2}/\alpha$ in (3), ensures that $F(d)$ is bounded by $2/3$ as long as m remains below $48\alpha^2$. Thus $F(d)$ is bounded by $2/3$ for all unmatched times until we reach segment $48\alpha^2$. This implies that if the sequence reaches segment $48\alpha^2$, then the regret with respect to the best expert will be at least $48\alpha^2\sqrt{T}/(16\alpha)(1/3) = \alpha\sqrt{T}$ which contradicts the fact that A is a $\alpha\sqrt{T}$-regret algorithm, so it cannot be the case that the sequence has $48\alpha^2$ or more segments. □

Define a *monotone* f-compliant algorithm to be an f-compliant algorithm A such that when *GenerateBadSeq* is applied to A it is the case that for all m and m', for all $t \in \mathcal{T}_m$ and $t' \in \mathcal{T}_{m'}$, if $m < m'$ then $t < t'$. In other words, an f-compliant algorithm is monotone if every m segment consists of a contiguous set of time steps. The following observation is useful in simplifying the proof, allowing us to further restrict our attention to the class of monotone f-compliant algorithms. It says that a lower bound on the performance of monotone algorithms will imply the general lower bound.

Lemma 7 *Consider any non-monotone f-compliant algorithm A, and let $\mathbf{g} = GenerateBadSeq(A, f, \gamma)$. There exists a monotone f-compliant algorithm A' with $\mathbf{g}' = GenerateBadSeq(A', f, \gamma)$ such that*

$$\sum_{t=1}^{T} g'_{A',t} > \sum_{t=1}^{T} g_{A,t}.$$

Proof If A is not monotone, then there must be some time step t and some distance $d > 0$ such that $m(d+2) = m(d+1)+1$ and $|d_t| = d$, $|d_{t+1}| = d+1$, $|d_{t+2}| = d+2$, and $|d_{t+3}| = d+1$. Here the first crossover into the $m(d+2)$ segment occurs at time $t+2$, and we cross back into the $m(d+1)$ segment at time $t+3$. Since A is f-compliant,

$$g_{A,t+1} + g_{A,t+2} + g_{A,t+3} = F(d) + F(d+1) + (1 - F(d+2)) = F(d) + 1 - f(d+2).$$

Now, consider a modified f-compliant algorithm A' that is the same as A everywhere except it chooses to have the weight it places on the leading expert at time $t+2$ treated as arbitrarily close to but *greater than* $1/2 + F(d+1)$ instead of arbitrarily close to but *less than* $1/2 + F(d+1)$, and sets the weight of this expert at time $t+3$ arbitrarily close to but less than $1/2 + F(d)$. This has the effect of modifying the sequence of distances so that $|d_{t+2}| = d$; the rest of the sequence remains the same. On this modified sequence,

$$g'_{A',t+1} + g'_{A',t+2} + g'_{A',t+3} = F(d) + (1 - F(d+1)) + F(d) = F(d) + 1 - f(d+1).$$

Since $m(d+2) > m(d+1)$, it must be the case that $f(d+2) > f(d+1)$ and the total gain of A' is strictly higher than the total gain of A.

If A' is not monotone, this transformation process can be repeated until a monotone f-compliant algorithm is found. Each time, the gain of the algorithm will strictly increase, yielding the result. □

The above lemma shows how we can change an f-compliant algorithm into a monotone f-compliant algorithm whose performance is at least as good. Therefore, we can consider only monotone algorithms.

We are now ready to prove the main lower bound theorem.

Proof of Theorem 6 First, consider the case in which the main `while` loop of *GenerateBadSeq*(A, f, γ) terminates before time T. It must be the case that $G_{avg,t-1} - G_{A,t-1} > 0.115\sqrt{T}/\gamma = \Omega(\sqrt{T})$ and there is nothing more to prove.

Throughout the rest of the proof, assume that the main `while` loop is never exited while generating the sequence $\mathbf{g}$. From Lemma 5 we know that if there are at least $T/(128\alpha^2)$ pairs of matched times in the m segment, then the loss to the average from these times will cancel the gain from unmatched times in this segment. By Lemma 6 there are at most $48\alpha^2$ segments. If the algorithm has *exactly* $T/(128\alpha^2)$ pairs of matched times at each segment, it will have at most a total of $T/(128\alpha^2)(48\alpha^2) = (3/8)T$ pairs of matched times and will cancel all of its gain over the average from the unmatched times in all segments. Note that there are at most $48\alpha^2\sqrt{T}/(16\alpha) = 3\alpha\sqrt{T}$ unmatched times. Since we have chosen T such that $\alpha < \sqrt{T}/150$, we can bound this by $0.02T$. This implies that there are at least $0.49T$ pairs of matched times. We define the following quantity for algorithm A: $x_m = |\mathcal{M}_m|/2 - T/(128\alpha^2)$. We have that

$$\sum_{m=1}^{48\alpha^2} x_m = \left(\sum_{m=1}^{48\alpha^2} \frac{|\mathcal{M}_m|}{2}\right) - \frac{3T}{8} \geq 0.49T - (3/8)T = 0.115T.$$

Let m^* be the first segment for which we have $\sum_{i=1}^{m^*} x_i \geq 0.115T$. Since we consider only monotone algorithms we know that by that time no segments larger than m^* have been visited. For every k, $1 \leq k \leq m^*$, we have $z_k = \sum_{i=k}^{m^*} x_i > 0$ (otherwise m^* would not be the first segment). Note that we can bound the regret to the average from below as follows,

$$\begin{aligned}
\sum_{i=1}^{m^*} x_i \frac{2^{i-1}}{\gamma\sqrt{T}} &= \frac{1}{\gamma\sqrt{T}} x_1 + \frac{1}{\gamma\sqrt{T}} \sum_{i=2}^{m^*} x_i \left(1 + \sum_{j=1}^{i-1} 2^{j-1}\right) \\
&= \frac{1}{\gamma\sqrt{T}} \sum_{i=1}^{m^*} x_i + \frac{1}{\gamma\sqrt{T}} \sum_{i=2}^{m^*} \sum_{j=2}^{i} 2^{j-2} x_i \\
&= \frac{1}{\gamma\sqrt{T}} z_1 + \frac{1}{\gamma\sqrt{T}} \sum_{j=2}^{m^*} 2^{j-2} z_j \geq \frac{0.115T}{\gamma\sqrt{T}} = \frac{0.115\sqrt{T}}{\gamma}.
\end{aligned}$$

This shows that the regret to the average must be at least $0.115\sqrt{T}/\gamma = \beta\sqrt{T}$ where $\beta = 0.115\alpha/2^{48\alpha^2}$, yielding the first result of the theorem.

Finally, for any $\alpha' \leq 1/10$, let $\alpha = \alpha'\sqrt{\log T} \leq \sqrt{\log T}/10$. From the previous result, this implies that if the regret to the best expert is bounded by $\alpha'\sqrt{T \log T} = \alpha\sqrt{T}$, then the regret to the average must be at least $(0.115\alpha/2^{(48/100)\log T})\sqrt{T} = 0.115\alpha T^{1/2-48/100} = \Omega(T^{1/50})$. This proves the second part of the theorem. □

6 Open questions

The results in this paper depend heavily on the fact that the gain of the algorithm at each time step is the weighted average of the gains of the experts. This can be interpreted as the expected gain that the algorithm would receive if it chose a single expert to follow on each time step according to its current distribution and subsequently received the gain of this expert. We might instead consider a scenario in which the algorithm is able to combine the advice of the experts in more sophisticated ways and receive a gain based on this combination, for example based on the squared loss of the combined prediction. It is not clear if similar results could be proved in such a setting. It would also be interesting to determine whether or not similar trade-offs exist in the portfolio setting (Cover 1991; Helmbold et al. 1998).

It is currently unknown whether or not it is possible to strengthen Theorem 6 to say that any algorithm with regret $O(\sqrt{T \log T})$ to the best expert must have regret $\Omega(T^{\epsilon})$ to the average for some constant $\epsilon > 0$. Such a result would further close the gap between our positive and negative results.

Acknowledgements We are grateful to Tong Zhang for sharing with us his insight on the modification of Prod in Sect. 4.3, to Manfred Warmuth and Andrew Barron for their thought-provoking remarks on the results presented here, and to the anonymous reviewers for many helpful comments, especially on the organization of the paper.

References

Auer, P., Cesa-Bianchi, N., & Gentile, C. (2002). Adaptive and self-confident on-line learning algorithms. *Journal of Computer and System Sciences*, *64*, 48–75.

Cesa-Bianchi, N., & Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge: Cambridge University Press.

Cesa-Bianchi, N., Mansour, Y., & Stoltz, G. (2007). Improved second-order bounds for prediction with expert advice. *Machine Learning*, *66*(2/3), 321–352.

Cover, T. (1991). Universal portfolios. *Mathematical Finance*, *1*(1), 1–19.

Even-Dar, E., Kearns, M., Mansour, Y., & Wortman, J. (2007). Regret to the best versus regret to the average. In *Twentieth annual conference on learning theory* (pp. 233–247).

Freund, Y. (2003). Predicting a binary sequence almost as well as the optimal biased coin. *Information and Computation*, *182*(2), 73–94.

Helmbold, D., Schapire, R., Singer, Y., & Warmuth, M. (1998). On-line portfolio selection using multiplicative updates. *Mathematical Finance*, *8*(4), 325–347.

Kalai, A., & Vempala, S. (2005). Efficient algorithms for on-line optimization. *Journal of Computer and System Sciences*, *71*(3), 291–307.

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

Vovk, V. (1998). A game of prediction with expert advice. *Journal of Computer and System Sciences*, *56*(2), 153–173.

Zhang, T. (2007). Personal communication.