Regret to the Best vs. Regret to the Average

Eyal Even-Dar[1], Michael Kearns[1], Yishay Mansour[2*], Jennifer Wortman[1]

[1] Department of Computer and Information Science, University of Pennsylvania
[2] School of Computer Science, Tel Aviv University

Abstract. We study online regret minimization algorithms in 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 fixed mixture of experts, and 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 $\Omega(T)$. We then describe and analyze a new multi-phase algorithm, which achieves cumulative regret only $O(\sqrt{T} \log T)$ to the best expert and has only *constant* regret to any fixed distribution over experts (that is, with no dependence on either T or the number of experts N). The key to the new algorithm is the gradual increase in the "aggressiveness" of updates in response to observed divergences in expert performances.

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,

[*] 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.

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) gain at each time step. The gain of the algorithm on a given time step is then a weighted average of these expert gains. The regret of the algorithm is measured in terms of cumulative gains over time. Our results establish hard 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 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 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 $1, 0, 1, 0, \cdots$, while the gains for expert 2 are $0, 1, 0, 1, \cdots$. Typical regret minimization algorithms (such as Exponential Weights [1, 2], Follow the Perturbed Leader [3], and the Prod algorithm [4]) will yield a gain of $T/2 - O(\sqrt{T})$, 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 $c/\sqrt{T}$; and therefore in odd steps the probability of expert 2 is only $(1/2 - c/\sqrt{T})$

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)
BestAverage	$O(\sqrt{TN} \log T)$	O(1)	O(1)
BestWorst	$O(N\sqrt{T \log N})$	$O(\sqrt{T \log N})$	0
EW	$O(T^{\frac{1}{2}+\alpha} \log N)$	$O(T^{\frac{1}{2}-\alpha})$	$O(T^{\frac{1}{2}-\alpha})$

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

(where the value of c depends on the specific algorithm). Note that adding a third expert, which is the average, would not change this.[3]

This paper establishes a sequence of results that demonstrates 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. 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 $N = 2$.

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), 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 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. Remaining closer to the specifics of existing algorithms, we also show that by restarting the Exponential Weights algorithm with progressively more aggressive learning rates (starting initially at the most conservative rate of $1/T$), we achieve a somewhat inferior tradeoff of $O(\sqrt{TN} \log T)$ regret to the best expert and constant regret to the average.

Our algorithms are somewhat different than many of the traditional regret minimization algorithms, especially in their apparently essential use of *restarts*

[3] 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.

that are driven by observed differences in expert performance. We show that this 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).

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. Suppose D_1 places all weight on one expert, while D_2 places all weight on a second. Competing simultaneously with both distributions is then equivalent to competing with the best expert.

Finally, we remark that our lower bounds on the trade-off between best and average regret *cannot* be circumvented by simply adding an "average expert" and running standard no-regret algorithms, even with the use of a prior distribution with a significant amount of weight on the average.[4]

Related Work: Previous work by Auer et al. [5] 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. Vovk [6] 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

In the classic experts framework, each expert $i \in \{1, \cdots, N\}$ receives a gain $g_{i,t} \in [0,1]$ at each time step t.[5] 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 at 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 $G_{D,t} = \sum_{i=1}^{N} D(i) G_{i,t}$.

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 $R_{best,A,T} = \max\{G_{best,T} - G_{A,T}, 1\}$. [6] In this work, we

[4] Achieving a constant regret to the average would require a prior of $1 - O(1/T)$ on this artificial expert and a learning rate of $O(1/T)$. Putting this much weight on the average results in $\Omega(T)$ regret to each of the original experts.

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

[6] 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.

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 an arbitrary distribution D, which is defined similarly.

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 $\Theta(T)$.

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 [1, 2], Follow the Perturbed Leader [3], and the Prod algorithm [4] are all examples of difference algorithms.[7] 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 $p_{1,1} \leq 1/2$.

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 $p_{1,t} \geq 2/3$. 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.[8]

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 $0, 1, 0, 1, \cdots$. 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

[7] For Prod, this follows from the restriction on the instantaneous gains to $\{0, 1\}$.

[8] If such a ρ does not exists, then $R_{best,A,T} = \Omega(T)$ and we are done.

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 $\Omega(\rho)$. 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 [2] is a difference algorithm achieving the trade-off described in Section 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. This piece of the analysis is standard (see, for example, Theorem 2.4 in [7]), and gives us the following: $G_{best,T} - G_{EW,T} \leq \eta G_{EW,T} + \ln N/\eta$.

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

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

Together with the analysis in [7], 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.

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

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 (1). 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 via Restarts

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 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.

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 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. 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.

```
BestAverage ($G^*$)
Let $k^* = \frac{1}{2}\log G^* - 3$ and $\ell = 8\sqrt{NG^*}$
for $k = 1 : k^* - 1$ do
    Reset weights and run a new instance of EW($\eta$) with $\eta = \eta_k = \frac{2^k}{G^*}$ until a
    time $t$ such that $(G^p_{best,t} - G^p_{avg,t}) \geq \ell$
end
Reset and run EW($\eta$) with $\eta = \eta_{k^*} = \frac{2^{k^*}}{G^*}$ until time $T$
```

Fig. 2. The *BestAverage* algorithm for N experts.

Theorem 3. *Let A be a regret minimization algorithm with regret at most R to the best expert and let BW be* BestWorst(A). *Then $R_{best,BW,T} = O(NR)$ 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.

4.2 An EW-Based Algorithm with Restarts

In Section 4.3 below we will give a general algorithm whose specialization will produce our best bicriteria regret bounds to the best and average. For pedagogical purposes, in this section we first present an algorithm for which we can prove inferior bicriteria bounds, but which works by directly applying restarts with increasingly aggressive learning rates to an existing difference algorithm. This multi-phase algorithm, which we shall call *BestAverage*, competes with the best expert while maintaining only constant regret to the average.

The algorithm is given in Figure 2. In each phase, a new instance of EW is run with a new, increasingly large value for the learning rate η. In the pseudocode and throughout the remainder of this paper, we will use the notation $G^p_{i,t}$ to mean the cumulative gain of expert i at time t only from the current phase of the algorithm, i.e. the amount that expert i has gained since the last time the learning rate η was reset. Similarly we will use $G^p_{best,t}$, $G^p_{avg,t}$, and $G^p_{BA,t}$ to be the gain of the best expert, average, and the *BestAverage* algorithm in the current phase through time t.

The following theorem states that *BestAverage* can guarantee a regret to the best expert that is "almost" as low as a standard no-regret algorithm while maintaining a constant regret to the average. The proof, which involves an analysis of the algorithm's gain compared to the gain of the best expert and the average both in the middle and at the end of each phase, has been omitted due to lack of space. The main insight of the proof is that whenever *BestAverage* exits a phase, it must have a quantifiable gain over the average. While the algorithm may lose to the average during the next phase, it will never lose much more than the gain it has already acquired. At the same time, we can bound how far the average is

```
**PhasedAggression** $(A, R, D)$
**for** $k = 1 : \log(R)$ **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 it distribution
        Use $p_t = \eta q_t + (1 - \eta)D$
    **end**
**end**
Reset and run a new instance of $A$ until time $T$
```

Fig. 3. The *Phased Aggression* algorithm for N experts.

behind the best expert at any given phase and use this to bound the regret of the algorithm to the best expert.

Theorem 4. *Let* $G^* \leq T$ *be an upper bound on* G_{max}*. Then* $R_{best,BA,T} = O(\sqrt{G^* N} \log G^*)$ *and* $R_{avg,BA,T} \leq 2$*.*

This theorem can be extended to hold when we would like to compete with arbitrary distributions D by using the generalized version of EW with prior D. The term $\sqrt{N}$ in the regret to the best expert will be replaced by $\max(\sqrt{N}, \max_{i \in N} \ln(1/D(i)))$.

4.3 Improved Dependence on N and Fixed Mixtures

Figure 3 shows *Phased Aggression*, an algorithm that achieves similar guarantees to *BestAverage* with a considerably better dependence on the number of experts. This algorithm has the added advantage that it 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. We note that this algorithm (or reduction) is similar in spirit to the EW-based approach above, in the sense that each successive phase is moving weight from something that is not learning at all (the fixed distribution D) to an algorithm that is implicitly learning aggressively (the given algorithm A). As before, new phases are invoked only in response to greater and greater outperformance by the current best expert.

Theorem 5. *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

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

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 $G^p_{best,t} - G^p_{D,t} > 2R$. 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. □

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 obliterated 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 *Phased Aggression* algorithm 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 will 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 $\boldsymbol{g}$ of length T such that if $R_{best,A,T} \leq \alpha\sqrt{T}$ then $R_{avg,A,T} \geq \beta\sqrt{T}$. Additionally, for any constant $\alpha' > 1/10$ 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 $\boldsymbol{g}$ 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 full proof is rather technical and appears in a separate subsection below.

5.1 Proof of Theorem 6

Fix a constant $\alpha > 0$. Given an algorithm A, we will generate a sequence of expert gains $\boldsymbol{g}$ of length $T > (150\alpha)^2$ such that $\boldsymbol{g}$ will be "bad" for A. In Figure 4, we show how to generate such a sequence. Here d_t is the difference between the gains of the two experts at time t, and ϵ_t is the increase in the probability that the algorithm assigns to the current best expert since the last time the d_t was smaller. This is used 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 will be defined later in the analysis.

We say that an algorithm A is *f-compliant* if at any time t we have $\epsilon_t = f(d_{t-1}) \pm \delta$, for an arbitrarily small δ, and if for any time t in which $d_t = 0$, we have $p_{1,t} = p_{2,t} = 1/2$. For the sake of this analysis, it is more convenient to think of ϵ_t as being exactly equal to $f(d_{t-1})$ and to allow the algorithm to "choose" whether it should be considered larger or smaller. Lemma 1 states that given the sequence generation process in Figure 4, we can concentrate only on the class of f-compliant algorithms. Due to space constraints, the proof is omitted.

Lemma 1. *Consider any algorithm A and let $\boldsymbol{g} = GenerateBadSeq(A, f, \gamma)$. There exists an f-compliant algorithm A' such that $GenerateBadSeq(A', f, \gamma) = \boldsymbol{g}$ and at any time t, $g_{A',t} \geq g_{A,t}$.*

Given an f-compliant algorithm, we can write its probabilities as a function of the difference between expert gains d_t. We define a function $F(d) = 1/2 +$

```
GenerateBadSeq($A, f, \gamma$)
$t = 1$; $G_{avg,0} = G_{A,0} = d_0 = 0$;
while ($G_{avg,t-1} - G_{A,t-1} \le 0.115\sqrt{T}/\gamma$) do
    $p_{1,t} = A(\boldsymbol{g}); p_{2,t} = 1 - A(\boldsymbol{g})$
    if ($d_{t-1} = 0$) then
        if $\left(p_{1,t} \le \frac{1}{2}\right)$ 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
        $i_t = \mathrm{argmax}_i\, G_{i,t}; j_t = \mathrm{argmin}_j\, G_{j,t};$
        $last(|d_{t-1}|) = p_{i_t,t};$
        $\epsilon_t = p_{i_t,t} - last(|d_{t-1} - 1|);$
        if ($\epsilon_t \le f(|d_{t-1}|)$) 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. 4. Algorithm for creating a bad sequence for any algorithm A.

$\sum_{i=1}^{|d|} f(i)$, 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 $F(d_t)$ is an f-compliant algorithm.

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

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

The following fact is immediate from this definition and will be useful many times in our analysis.

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

We define the (possibly noncontiguous) *m segment* to be the set of all times t for which $m(d_t) = m$, or more explicitly, all times t for which $(m - 1)(\sqrt{T}/(16\alpha)) \le |d_t| < m(\sqrt{T}/(16\alpha))$. We denote this set of times by $\mathcal{T}_m$.

We now 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 $d+1$ at 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. 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 generally 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 algorithm's regret to the best expert and the average of all experts for each pair of matched times.

Lemma 2. *For any f-compliant algorithm A and any pair of matched times t_1 and t_2 in the m segment, the algorithm's gain from times t_1 and 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 3. *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., $\sum_{t\in\mathcal{UM}_m} g_{A,t} - 1/2 \leq 2^m\sqrt{T}/(256\gamma\alpha^2)$.*

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 (2), it follows that

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

□

Combining lemmas 2 and 3, 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 4. *For any given 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. $\sum_{t\in\mathcal{T}_m} g_{A,t} \leq \sum_{t\in\mathcal{T}_m}(1/2) - 2^{m-1}x/(\gamma\sqrt{T})$.*

Proof. From Lemma 2, the loss of A with respect to the average for each pair of matched times is $2^{m-1}/(\gamma\sqrt{T})$. From Lemma 3, A could not have gained more than $2^m\sqrt{T}/(256\alpha^2\gamma)$ over the average in the m segment. Since there are at least $2T/(128\alpha^2) + 2x$ 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 5. *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 $\boldsymbol{g} = GenerateBadSeq(A, f, \gamma)$.*

Proof. Once again we assume that leading expert is expert 1. Setting $\gamma = 2^{48\alpha^2}/\alpha$ in (2), 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. □

The following observation will be useful in simplifying the main proof, allowing us to further restrict our attention to the class of *monotone* f-compliant algorithms, where an algorithm is *monotone* if it never returns to a segment m after moving on to segment $m + 1$. A lower bound on the performance of monotone algorithms will imply the general lower bound.

Lemma 6. *Suppose $d_t = d > 0$, $d_{t+1} = d + 1$, $d_{t+2} = d + 2$, and $d_{t+3} = d + 1$. The gain of an f-compliant algorithm will not decrease if we instead let $d_{t+2} = d$.*

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

Proof. (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 $\boldsymbol{g}$. From Lemma 4 we know that if there are at least $T/(128\alpha^2)$ pairs of matched times in the ℓ segment, then the loss to the average from these times will cancel the gain from unmatched times in this segment. By Lemma 5 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 need not worry about timing issues). 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 as follows.

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

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.

If we now let T be large enough that $\alpha \leq \sqrt{\log T}/10$, this regret must be at least $(0.115\alpha/2^{(48/100)\log T})\sqrt{T} = 0.115\alpha T^{1/2-48/100} = O(T^{1/50})$, which proves the last part of the theorem. □

Acknowledgments

We are grateful to Manfred Warmuth and Andrew Barron for their thought-provoking remarks on the results presented here.

References

1. N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.
2. Y. Freund. Predicting a binary sequence almost as well as the optimal biased coin. *Information and Computation*, 182(2):73–94, 2003.
3. A. Kalai and S. Vempala. Efficient algorithms for on-line optimization. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.
4. N. Cesa-Bianchi, Y. Mansour, and G. Stoltz. Improved second-order bounds for prediction with expert advice. In *COLT*, pages 217–232, 2005.
5. P. Auer, N. Cesa-Bianchi, and C. Gentile. Adaptive and self-confident on-line learning algorithms. *Journal of Computer and System Sciences*, 64:48–75, 2002.
6. V. Vovk. A game of prediction with expert advice. *Journal of Computer and System Sciences*, 56(2):153–173, 1998.
7. N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games.* Cambridge University Press, 2006.