RJ10418 (A0710-007) October 5, 2007
Computer Science

IBM Research Report

Adaptive Algorithms for Online Decision Problems

Elad Hazan
IBM Research Division
Almaden Research Center
650 Harry Road
San Jose, CA 95120-6099

C. Seshadhri
Princeton University
35 Olden Street
Princeton, NJ 08540
(work done at IBM Almaden Research Center)

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Adaptive Algorithms for Online Decision Problems

Elad Hazan
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120
hazan@us.ibm.com

C. Seshadhri *
Princeton University
35 Olden St.
Princeton, NJ 08540
csesha@cs.princeton.edu

Abstract

We study the notion of learning in an oblivious changing environment. Existing online learning algorithms which minimize *regret* are shown to converge to the *average* of all locally optimal solutions. We propose a new performance metric, strengthening the standard metric of regret, to capture convergence to locally optimal solutions, and propose efficient algorithms which provably converge at the optimal rate.

One application is the portfolio management problem, for which we show that all previous algorithms behave suboptimally under dynamic market conditions. Another application is online routing, for which our adaptive algorithm exploits local congestion patterns and runs in near-linear time. We also give an algorithm for the tree update problem that is statically optimal for every sufficiently long contiguous subsequence of accesses.

Our algorithm combines techniques from data streaming algorithms, composition of learning algorithms, and a twist on the standard experts framework.

1 Introduction

In online optimization the decision maker sequentially chooses a decision without knowledge of the future, and pays a cost based on her decision and the observed outcome. The game theory and machine learning literature has produced a host of algorithms which perform nearly as well as the best single decision in hindsight. Formally, the average *regret* of the online player, which is the average difference between her cost and the cost of the best strategy in hindsight, approaches zero as the number of game iteration grows. Examples of online optimization scenarios for which such online algorithms were successfully applied include portfolio management [Cov91], online routing [TW03], and boosting [FS97].

Low regret algorithms are particularly useful in scenarios in which the environment variables are sampled from some (unknown) distribution. In such cases, low regret algorithms effectively "learn" the environment and approach the optimal strategy. However, if the underlying distribution changes, no such claim can be made. Indeed, we later describe simple examples in which low regret algorithms do not converge to the locally optimal strategy.

For example, consider the case of the portfolio management problem. If the stock price changes are sampled according to a certain distribution, Cover showed that low regret algorithms converge to the optimal strategy (in this case a *constant rebalanced portfolio*). However, if the market shifts to a different distribution, such a guarantee cannot be proved.

*This work was done while the author was a research intern at the IBM Almaden Research Center

Intuitively, the reason is that all low regret algorithms for portfolio management "remember" the entire market history which can be largely irrelevant if the underlying distribution changes. Similarly, in online routing we would like our algorithm to adapt to different network congestion scenarios and approach the optimum corresponding to the current congestion, rather than the long-term aggregated congestion.

In this paper we address this question of adapting to a changing environment. We argue that the correct measure of performance is Adaptive-Regret, or regret on any interval of history. If an online algorithm has low regret on *every* interval in history, then intuitively it will converge to the local optimum for each interval, and hence successfully track environment changes.

We give an efficient generic scheme for converting any low regret algorithm into a low Adaptive-Regret algorithm. Building on existing algorithms, we propose online optimization algorithms with nearly optimal Adaptive-Regret for portfolio management, online routing, tree updates, and more general settings. For the case of online routing, the algorithm effectively exploits the structure of the problem to allow for efficient implementation (despite the fact that there exists exponentially many paths).

Our techniques include twists on the Multiplicative Weights algorithm from the learning community as well as application of results from the data-streaming literature (as far as we know, for the first time in learning-theoretic applications).

It has come to our notice that independently in the information theory community some related work was done in [KS07b, KS07a]. The techniques used are completely different, and our setting is more general (i.e. the referenced papers do not to deal with general convex loss functions). In addition, our algorithms are more efficient.

1.1 Our Results

In an online decision problem, in each round $t = 1, 2, ...$, the decision maker plays a point x_t from a convex domain $K \subseteq \mathbb{R}^n$. A loss function f_t is presented, and the decision maker incurs a loss of $f_t(x_t)$. The standard performance measure is regret, which is the difference between the loss incurred by the online player using algorithm $\mathcal{A}$ and the best fixed optimum in hindsight:

$$\text{Regret}_T(\mathcal{A}) = \sum_{t=1}^{T} f_t(x_t) - \min_{x^* \in K} \sum_{t=1}^{T} f_t(x^*)$$

We consider an extension of the above quantity to measure the performance of a decision maker in a changing environment:

Definition 1.1. *The Adaptive-Regret of an online convex optimization algorithm $\mathcal{A}$ is defined as the maximum regret it achieves over any contiguous time interval. Formally*

$$\textit{Adaptive-Regret}_T(\mathcal{A}) \triangleq \sup_{I=[r,s] \subseteq [T]} \left\{ \sum_{t=r}^{s} f_t(x_t) - \min_{x^* \in K} \sum_{t=r}^{s} f_t(x^*) \right\}$$

Obviously Adaptive-Regret is a strict generalization of regret. Intuitively, an algorithm with $O(R)$ Adaptive-Regret converges to the locally optimal solution in each interval of length $\Omega(R)$. In the following sections we propose and analyze algorithms which attain Adaptive-Regret bounds for a variety of problems. These Adaptive-Regret bounds match the lower bounds for regular regret up to logarithmic factors. In addition, the most efficient

version of our algorithms have only a logarithmic running time overhead over the most efficient known algorithms. We call this class of algorithms *Follow-The-Leading-History* (FLH). There are broadly two versions of FLH, one for exp-concave functions and one of general convex functions. For ease of notation, we will refer to them both as FLH, as the version will be clear by the context. Furthermore, FLH has an advanced implementation (called AFLH) which has slightly worse Adaptive-Regret guarantees but much better running time.

Throughout the paper the O-notation hides absolute constants.

Theorem 1.2. *Suppose the functions $f_1, \cdots, f_T$ are α-exp concave (for some constant α) and there exists an algorithm giving $R(T)$ regret with running time V. The running time of algorithm FLH is $O(VT)$ and Adaptive-Regret$_T(FLH) \leq R(T) + O(\frac{1}{\alpha}\log T)$. The running time of $AFLH$ is $O(V \log T)$ and Adaptive-Regret$_T(AFLH) \leq R(T)\log T + O(\frac{1}{\alpha}\log^2 T)$.*

For general convex loss functions, we get a similar theorem -

Theorem 1.3. *Suppose the functions $f_1, \cdots, f_T$ are convex and bounded by $f(x) \in [0, M]$ in the convex set and there exists an algorithm giving $R(T)$ regret with running time V. The running time of algorithm FLH is $O(VT)$ and Adaptive-Regret$_T(FLH) \leq R(T) + O(M\sqrt{T\log T})$. The running time of $AFLH$ is $O(V \log T)$ and Adaptive-Regret$_T(AFLH) \leq R(T)\log T + O(M\sqrt{T\log^3 T})$.*

For convex functions, we actually prove a slightly stronger statement, where we get tradeoffs between a multiplicative factor over the optimal loss and an additive error. In order to streamline the exposition, we defer the formal statement of results concerning the applications of the above theorems to section 2.

To motivate our new measure of performance, we give an example in which current algorithms behave suboptimally, and explain how low Adaptive-Regret algorithms overcome these problems.

Suboptimal behaviour of existing algorithms. Consider the online convex optimization framework, in which the decision maker chooses a point from the subset of the real line $x_t \in [-1, 1]$. The convex loss functions are $f_t(x) = (x-1)^2$ for the first $T/2$ iterations, and $f_t(x) = (x+1)^2$ in the last $T/2$ iterations. For the sake of simplicity, consider the "follow-the-leader" (FTL) algorithm which at each iteration predicts the minimum of the aggregated loss function thus far (this algorithm is known to attain $O(\log T)$ regret for this setting [HKKA06]).

The strategy chose by this algorithm will be $x_t = 1$ in the first $T/2$ iterations, and then slowly shift towards $x_t = 0$ in the last $T/2$ iterations. Despite the fact that the total regret is $O(\log T)$, it is easy to see that the Adaptive-Regret is $\Omega(T)$ in the last $T/2$ iterations (where the optimum is obviously -1).

Although we have considered the FTL algorithm, all known logarithmic regret algorithms (Online Newton Step, Cover's algorithm, Exponential Weighting) will behave similarly. In addition, although we described the simplest setting, the same issue arises in the portfolio management and online shortest paths problems described below.

In contrast, our algorithms which attain $O(\log T)$ Adaptive-Regret, initially predict 1, and at $T/2$ start shifting towards -1, and complete this shift at $T/2 + O(\log T)$, thus behaving locally optimal.

1.2 Relation to previous work

The most relevant previous work are the papers of Herbster and Warmuth [HW98] and Bousquet and Warmuth [BW03] on "tracking the best expert". The focus of their work was on the discrete expert setting and exp-concave loss functions. In this scenario, they proved regret bounds versus the best *k-shifting* expert, where the optimum in hindsight is allowed to change its value k times. Singer [Sin] looked at portfolio management, and gave an algorithm that is competitive with strategies that switch between different (single) assets.

Our setting differs from this expert setting in several respects. First, we generally consider continuous decision sets rather than discrete. Although it is possible to discretize continuous sets (i.e. the simplex for portfolio management) and apply previous algorithms, such reductions are inefficient. Presumably it might be possible to apply random walk techniques such as Kalai and Vempala [KV03], but that too would be far less efficient than the techniques presented hereby.

For the discrete problems we consider (i.e. online routing), the loss functions are typically linear rather than exp-concave as in [HW98]. Zinkevich's gradient descent algorithm [Zin03] can be shown to attain near optimal Adaptive-Regret, however it is again not clear how to do so efficiently for structured problems.

As for performance guarantees, it is easy to see that our notion of Adaptive-Regret generalizes (and is not equivalent to) regret versus the best *k-shifting* optimum. We also remark that the techniques we use are quite different that previous approaches. One component is inspired by [HW98] of using Multiplicative Weights to obtain Adaptive-Regret bounds for shifting experts. However, the set of experts in our setting are other algorithms rather than strategies, and its composition and size changes during the run time of the main algorithm: experts are removed and added. The second major component is a sparsification of the expert set, which relies on data streaming techniques.

1.3 The Data-Streaming problem

Our efficient implementation uses an interesting twist on the standard experts scenario - usually, there is a fixed set of experts which learning algorithms track. In our situation, we have a set of experts that is dynamic and keeps changing. In each round, a new expert is brought in and some experts are removed. This is done to keep the number of experts small, and allows us to design efficient learning algorithms. Based on the structure and properties of experts, this removal of experts needs to be done delicately to ensure that regret guarantees are maintained. One of the interesting aspects of this work is the use of techniques from streaming algorithms are used for learning. We describe the main streaming problem that we need to deal with. The details of the connection between this to learning will be explained later on in the paper.

This problem can be stated without any mentioning of our learning algorithm, and is of independent interest. Suppose the integers $1, 2, \cdots$ are being "processed" in a streaming fashion. At time t, we have "read" the positive integers upto t and maintain a very small subset of them S_t. The set S_t should be well spread out in a exponential fashion (this will be explained precisely below). The sets are updated in a streaming manner - at time t, we have S_t and at $t+1$, we modify S_t to get S_{t+1}. The catch is that the only integer we can *add* to S_t to get S_{t+1} is $t+1$. We are free to remove whatever we wish. If some $i \leq t$ is not present in S_t, then it can never be in any $S_{t'}$, for $t' > t$. It is very natural to think of the

integers as data objects being streamed and our aim is to maintain a short "sketch" of the data seen so far. Once we discard a data item from our sketch, it is not possible to retrieve it.

Let us now precisely describe the conditions on the sets S_t.

Property 1.4. *1. For every positive $s \leq t$, $[s, (s+t)/2] \cap S_t \neq \phi$.*

2. For all t, $|S_t|$ is at most polylogarithmic in t.

3. For all t, $S_{t+1} \backslash S_t = \{t+1\}$.

A randomized procedure to construct these sets in a streaming fashion is given in [GJKK]. Woodruff [Woo07] gave an elegant deterministic solution which we describe in Appendix B.

2 Applications

Portfolio management. In the universal portfolio management setting, an online investor iteratively distributes her wealth on a set of N assets. After committing to this distribution p_t, the market outcome is observed in a form of a *price relatives vector* r_t and the investor attains utility of $\log(p_t \cdot r_t)$. This formulation is a special case of the online convex optimization setting with a (negative) logarithmic loss function, which is exp-concave. In his remarkable original paper, Cover [Cov91] analyzes an algorithm called UNIVERSAL which attains

$$\text{Regret}_T(\text{UNIVERSAL}) = \max_{p^*} \sum_{t=1}^{T} \log(p^* \cdot r_t) - \sum_{t=1}^{T} \log(p_t \cdot r_t) = O(n \log T)$$

This was shown to be tight up to constant factors [OC98]. As in our previous example above, it is easy to construct examples in which the Adaptive-Regret of Cover's algorithm is $\text{Adaptive-Regret}_T(\text{UNIVERSAL}) = \Omega(T)$. Using Theorem 1.2, we can prove -

Corollary 2.1. *There exists an online algorithm $\mathcal{A}$ that for any sequence of price relative vectors $r_1, ..., r_T$, produces wealth distributions $p_1, ..., p_T$ such that*

$$\textit{Adaptive-Regret}_T(\mathcal{A}) = O(n \log T)$$

Further, the running time of this algorithm is polynomial in n and T. The running time can be made polynomial in n and $\log T$ with the guarantee $\textit{Adaptive-Regret}_T(\mathcal{A}) = O(n \log^2 T)$

This bound matches the optimal bound on regular regret, and essentially gives the first theoretical improvement over Cover's algorithm [1].[2]

[1] We note that the running time of Cover's algorithm is exponential. Kalai and Vempala [KV03] gave a polynomial time implementation, which we use for this result. Building on the Online Newton algorithm [HKKA06], we can obtain not only poly-time, but running time which depends only logarithmically on the number of iterations T, albeit introducing dependency in the gradients of the loss functions.

[2] Our guarantee is not to be confused with Singer's "switching portfolios" [Sin]. In his paper Singer deals with switching between single assets, and not CRPs (which is stronger) as is in our case. Our approach is also more efficient.

Online routing and shortest paths. In the online shortest paths (OSP) problem, an online decision maker iteratively chooses a path in a weighted directed graph without knowing the weights in advance and pays a cost which is the length of her path. Let the graph have m edges, n vertices and weights in the range $[0, 1]$. Takimoto and Warmuth [TW03] gave a Multiplicative Weights algorithm which attains (optimal) regret of $O(\sqrt{T})$. Kalai and Vempala [KV05] showed how a simpler approach can give efficient algorithms for OSP and other structured graph problems. Both approaches yield the following more general guarantee:

$$\text{E[total weight]} \leq (1+\varepsilon)(\text{best weight in hindsight}) + \frac{O(mn \log n)}{\varepsilon}$$

Here best weight in hindsight refers to the total weight of the best single path. Both approaches suffer from the suboptimal behavior explained before, namely the online router may converge to the shortest path of the aggregated graph, which could be very different from the locally optimal path.

Based on the stronger version of Theorem 1.3, we can construct an algorithm with the following guarantee for *any* interval $I \subseteq [T]$

$$\text{E[total weight on } I] \leq (1+\varepsilon)(\text{best weight in hindsight on } I) + \frac{O(mn \log n \log T + n \log^2 T)}{\varepsilon}$$

Taking $\varepsilon = \frac{1}{\sqrt{Tmn \log^2 T \log n}}$ we obtain

Corollary 2.2. *For the OSP problem, there exists an algorithm $\mathcal{A}$ with running time polynomial in $m, n, \log T$ that guarantees -*

$$\textit{Adaptive-Regret}_T = O\big(\sqrt{Tmn \log^2 T \log n}\big)$$

This algorithm attains almost optimal Adaptive-Regret. Using FTL ideas the algorithms are easily and efficiently applied to the variety of graph problems considered in [TW03, KV05].

Between static and dynamic optimality for search trees. The classic tree update problem was posed by Sleator and Tarjan in [ST85]. The online decision maker is given a sequence of access requests for items in the set $\{1, ..., n\}$. Her goal is to maintain a binary search tree and minimize the total lookup time and tree modification operations. This problem was looked at from an online learning perspective in [BCK03, KV05]. An algorithm is *statically optimal* if the total time taken by this algorithm is comparable (upto a constant) to the best tree in hindsight. Splay trees are known to be statically optimal, with a constant factor of $3 \log_2 3$. Kalai and Vempala [KV05] gave an efficient statically optimal tree algorithm the strong guarantee of low regret (in particular, the constant factor is basically 1). More specifically, they give a randomized algorithm such that -

$$\text{E[cost of algorithm]} \leq (\text{cost of best tree}) + 2n\sqrt{nT}$$

For this, they use a version of the Follow-The-Leader (FTL) which does not give the stronger guarantee of low Adaptive-Regret. We can use our techniques to give such an algorithm, that is basically statically optimal for *every* large enough contiguous subsequence of accesses (note that splay trees are also statically optimal for every contiguous subsequence but the constant multiplicative factor is $3 \log_2 3$). We design a lazy version of our algorithm with the following properties -

Theorem 2.3. *Suppose that for all $x \in K$ and $t \in [T]$, $f_t(x) \in [0, M]$. Let $R(T)$ be an upper bound on the regret of some learning algorithm over a sequence of T functions. There exists a randomized algorithm $\mathcal{A}$, such that with high probability*[3]*, for any $\varepsilon > 0$ less than some sufficiently small constant -*

1. *Adaptive-Regret$_T(\mathcal{A}) \leq R(T) + O(\frac{M\sqrt{T \log T}}{\varepsilon})$*
2. *Throughout the running time of $\mathcal{A}$, $x_t \neq x_{t-1}$ at most εT times.*

We can formulate the tree update problem in the learning setting by considering a point of the domain as a tree. Since it takes $O(n)$ time to update one tree to another (any tree can be changed to another in $O(n)$ rotations), the total modification time is $O(\varepsilon nT)$. Setting $\varepsilon = ((\log T)/T)^{1/4}$, we get -

Corollary 2.4. *Let $a_1, \cdots, a_T$ be accesses made. There is a randomized algorithm $\mathcal{A}$ for the tree update problem that for any contiguous subsequence of queries $I = \{a_r, a_{r+1}, \cdots, a_s\}$, gives the following guarantee with high probability -*

$$cost_I(\mathcal{A}) \leq cost_I(\text{best tree for } I) + O(nT^{3/4}(\log T)^{1/4} + n\sqrt{nT})$$

Although the additive term is worse than that of [KV05], it is still significantly sublinear, and we get a very strong version of static optimality.

3 The basic method

In this section we discuss the basic method and apply it to exp-concave loss functions, such as the logarithmic function appearing in portfolio management, and to convex loss functions, useful for applications such as online routing. The two different families of loss functions require different techniques, and distinct Adaptive-Regret bounds are obtained.

3.1 Exp-concave loss functions

First we consider α-exp concave loss functions, i.e. functions f_t such that $e^{-\alpha f_t}$ is a convex function. The basic algorithm, which we refer to as *Follow-the-Leading-History* (FLH), is detailed in the figure below. The basic idea is to use many online algorithms, each attaining good regret for a different segment in history, and choose the best one using expert-tracking algorithms. The experts are themselves algorithms, each starting to predict from a different point in history. The meta-algorithm used to track the best expert is inspired by the Herbster-Warmuth algorithm [HW98]. However, our set of experts continuously changes, as more algorithms are considered. This is further complicated in the next section when some experts are also removed.

We henceforth prove the following theorem, which shows the strong Adaptive-Regret guarantees of FTHL -

Theorem 3.1. *Suppose that algorithms $\{E^r\}$ attain regret of $R(T)$ over any interval of length T, and have running time V. Then the running time of algorithm FLH is $O(VT)$ and Adaptive-Regret$_T(FLH) = R(T) + O(\frac{1}{\alpha} \log T)$.*

This theorem immediately follows from the following stronger performance guarantee:

[3] the probability of error is $< T^{-2}$

Follow-the-Leading-History

1. Let $E^1, ..., E^T$ be online convex optimization algorithms.
2. For each t, $v_t = (v_t^1, ..., v_t^t)$ is a probability vector in $\mathbb{R}^t$. Initialize $v_1^1 = 1$.
3. In round t, set $\forall j \leq t$, $x_t^j \leftarrow E^j(f_{t-1})$ (the prediction of the j'th algorithm). play $x_t = \sum_{j=1}^t v_t^{(j)} x_t^{(j)}$.
4. After receiving f_t, set $\hat{v}_{t+1}^{(t+1)} = 0$ and perform update for $1 \leq i \leq t$ -
$$\hat{v}_{t+1}^{(i)} = \frac{v_t^{(i)} e^{-\alpha f_t(x_t^{(i)})}}{\sum_{j=1}^t v_t^{(j)} e^{-\alpha f_t(x_t^{(j)})}}$$
5. Addition step - Set $v_{t+1}^{(t+1)}$ to $1/(t+1)$ and for $i \neq t+1$ -
$$v_{t+1}^{(i)} = (1 - (t+1)^{-1}) \hat{v}_{t+1}^{(i)}$$

Theorem 3.2. *For any interval $I = [r, s]$ in time, the algorithm FLH gives $O(\alpha^{-1}(\ln r + \ln |I|))$ regret with respect to the best optimum in hindsight for I.*

By assumption expert E^r gives $R(|I|)$ regret in the interval I (henceforth, the time interval I will always be $[r, s]$). We will show that FLH will be competitive with expert E^r in I. To prove Theorem 3.2, it suffices to prove the following lemma.

Lemma 3.3. *For any $I = [r, s]$, the regret incurred by FLH in I with respect to expert E^r is at most $\frac{2}{\alpha}(\ln r + \ln |I|)$.*

We first prove the following lemma, which gives bounds on the regret in any round.

Lemma 3.4. *1. For $i < t$, $f_t(x_t) - f_t(x_t^{(i)}) \leq \alpha^{-1}(\ln \hat{v}_{t+1}^{(i)} - \ln \hat{v}_t^{(i)} + 2/t)$*

2. $f_t(x_t) - f_t(x_t^{(t)}) \leq \alpha^{-1}(\ln \hat{v}_{t+1}^{(t)} + \ln t)$

Proof. Using the α-exp concavity of f_t -

$$e^{-\alpha f_t(x_t)} = e^{-\alpha f_t(\sum_{j=1}^t v_t^{(j)} x_t^{(j)})} \geq \sum_{j=1}^t v_t^{(j)} e^{-\alpha f_t(x_t^{(j)})}$$

Taking logarithm,

$$f_t(x_t) \leq -\alpha^{-1} \ln \sum_{j=1}^t v_t^{(j)} e^{-\alpha f_t(x_t^{(j)})}$$

Hence,

$$\begin{aligned} f_t(x_t) - f_t(x_t^{(i)}) &\leq \alpha^{-1}(\ln e^{-\alpha f_t(x_t^{(i)})} - \ln \sum_{j=1}^{t} v_t^{(j)} e^{-\alpha f_t(x_t^{(j)})}) \\ &= \alpha^{-1} \ln \frac{e^{-\alpha f_t(x_t^{(i)})}}{\sum_{j=1}^{t} v_t^{(j)} e^{-\alpha f_t(x_t^{(j)})}} \\ &= \alpha^{-1} \ln\big(\frac{1}{v_t^{(i)}} \cdot \frac{v_t^{(i)} e^{-\alpha f_t(x_t^{(i)})}}{\sum_{j=1}^{t} v_t^{(j)} e^{-\alpha f_t(x_t^{(j)})}}\big) \\ &= \alpha^{-1} \ln \frac{\hat{v}_{t+1}^{(i)}}{v_t^{(i)}} \end{aligned} \tag{1}$$

The lemma is now obtained using the bounds of Claim 3.5 below.

□

Claim 3.5. *1. For $i < t$, $\ln v_t^{(i)} \geq \ln \hat{v}_t^{(i)} - 2/t$*

2. $\ln v_t^{(t)} \geq -\ln t$

Proof. By definition, for $i < t$, $v_t^{(i)} = (1 - 1/t)\hat{v}_t^{(i)}$. Also, $v_t^{(t)} = 1/t$. Taking the natural log of both these inequalities completes the proof. □

We are now ready to prove Lemma 3.3. It will just involve summing up the regret incurred in each round, using Lemma 3.4. The main idea is the use the first bound for the first round of I and then use the second bound for the remaining rounds.

Proof. (Lemma 3.3) We are looking at regret in I with respect to an expert E^r.

$$\begin{aligned} \sum_{t=r}^{s} (f_t(x_t) - f_t(x_t^{(r)})) &= (f_r(x_r) - f_r(x_r^{(r)})) + \sum_{t=r+1}^{s} (f_t(x_t) - f_t(x_t^{(r)})) \\ &\leq \alpha^{-1}\big(\ln \hat{v}_{r+1}^{(r)} + \ln r + \sum_{t=r+1}^{s} (\ln \hat{v}_{t+1}^{(r)} - \ln \hat{v}_t^{(r)} + 2/t)\big) \\ &= \alpha^{-1}(\ln r + \ln \hat{v}_{s+1}^{(r)} + \sum_{t=r+1}^{s} 2/t) \end{aligned}$$

Since $\hat{v}_{s+1}^{(r)} \leq 1$, $\ln \hat{v}_{s+1}^{(r)} \leq 0$. This implies that the regret is bounded by $2\alpha^{-1}(\ln r + \ln |I|)$. □

3.2 General convex loss functions

Here, we explore the case of general convex loss functions. For the sake of simplicity assume that the loss functions f_t are bounded in the domain by $f_t(x) \in [0, M]$. Note that we cannot expect to obtain logarithmic Adaptive-Regret, as even standard regret is known to be lower bounded by $\Omega(\sqrt{T})$. Instead, we derive relative loss bounds, or competitive ratio bounds, and as a special case obtain $O(\sqrt{T} \log T)$ Adaptive-Regret bounds.

We employ the same algorithm as before, and choose our experts accordingly. There is a slight difference, because instead of taking a convex combination of the experts' predictions,

we choose experts according to the probability vector v_t. Expert E^i is chosen (in round t) with probability $v_t^{(i)}$.

Our experts will be any algorithm $\mathcal{A}$ that attains low regret (i.e. the Multiplicative Weights algorithm, Perturbed Follow the Leader). The FLH version for convex functions is almost the same, with a slight change to the multiplicative update rule.

After receiving f_t, perform update for $1 \leq i \leq t$ -

$$\hat{v}_{t+1}^{(i)} = \frac{v_t^{(i)} e^{-\eta_t f_t(x_t^{(i)})}}{\sum_{j=1}^{t} v_t^{(j)} e^{-\eta_t f_t(x_t^{(j)})}}$$

The learning rate η_t will be set according to the bounds we want for strong competitiveness. The main theorem of this section is -

Theorem 3.6. *If each expert is implemented by a learning algorithm guaranteeing $R(T)$ regret (for T rounds), then the FLH algorithm has Adaptive-Regret$_T(FLH) \leq R(T) + O(M\sqrt{T}\log T)$.*

This theorem immediately follows for the next lemma (proven in appendix).

Lemma 3.7. *Let $0 < \alpha < \frac{1}{4}$. For any interval $I = [r, s]$, if expert E^r incurs loss L on I, then by setting $\eta_t = -M^{-1}\log(1-\alpha)$, the loss incurred by FLH on I is at most $(1+\alpha)L + M\alpha^{-1}\ln s$.*

4 Efficient implementation

Our aim now is to implement the above algorithms efficiently and using little space. At time t, the present implementation of FLH stores all the experts $E^1, \cdots, E^t$ and has to compute weights for all of them. Let V denote an upper bound on the running time of each expert (for one round). The time taken is at least $O(Vt)$. It is natural to ask whether all these experts are necessary, since experts that are close to each other (starting from very close time steps) will make similar predictions. Is there a way to sparsify the set of experts and still maintain our strong regret bounds? We address this question, using techniques from streaming algorithms.

The implementation will not depend on whether we deal with exp-concave or general convex functions.

Theorem 4.1. *Consider the standard implementation of FLH and suppose it provides $R(T)$ regret for T rounds. Then AFLH that has expected $O(V\log T)$ running time (for every round) and has an expected regret of $O(R(T)\log T)$.*

For general convex functions, it is often the case that some learning algorithm $\mathcal{A}$ has a stronger guarantees than sublinear regret. Given the functions $f_1, \cdots, f_T$, let OPT denote the loss of the optimal point in hindsight and loss($\mathcal{A}$) be the loss of $\mathcal{A}$. Then, for sufficiently small ε -

$$\text{loss}(\mathcal{A}) \leq (1+\varepsilon)OPT + \varepsilon^{-1}c(T)$$

For such a situation, we can also prove a version of Theorem 4.1 giving similar tradeoffs. The proof of this is basically the same as that of Theorem 4.1.

Theorem 4.2. *Suppose there exists algorithm $\mathcal{A}$ with running time of V per round such that $loss(\mathcal{A}) \leq (1+\varepsilon)OPT + \varepsilon^{-1}c(T)$ for any sufficiently small ε. Consider the implementation of FLH (using $\mathcal{A}$ as experts) and let $loss_I(FLH)$ be the loss of FLH (using $\mathcal{A}$ as experts) on time interval $I = [r, s]$. Suppose $loss_I(FLH) \leq (1+\varepsilon)loss_I(E^r) + \varepsilon^{-1}d(T)$. The loss of algorithm AFLH in I is bounded by -*

$$loss_I(AFLH) \leq (1+\varepsilon)OPT(I) + O\Big(\frac{\log T}{\varepsilon}(c(T)+d(T))\Big)$$

The running time of AFLH is $O(V \log T)$.

We show that it suffices to store only $O(\log t)$ experts at time t. At time t, there is a working set S_t of experts. In the old implementation of FLH, this set can be thought of to contain $E^1, \cdots, E^t$. For the next round, a new expert E^{t+1} was added to get S_{t+1}. To decrease the sizes of these sets, the efficient implementation will also *remove* some experts. Once an expert is removed, it is never used again (it cannot be added again to the working set). The algorithm will perform the multiplicative update and mixing step only on the working set of experts. The working set of experts has a very dynamic behaviour, and we will ensure that it has small size. On the other hand, it will have enough experts to allow us to get low regret.

The algorithm AFLH works exactly the same as standard FLH, with the added *pruning* step. This is the step where certain experts are removed to update the new working set S_{t+1} for round $t+1$. We remind the reader of the properties of $S_t \subseteq [1, n]$ required -

1. For every positive $s \leq t$, $[s, (s+t)/2] \cap S_t \neq \phi$.
2. For all t, $|S_t|$ is at most polylogarithmic in t.
3. For all t, $S_{t+1} \backslash S_t = \{t+1\}$.

There is a randomized construction for these sets given by [GJKK]. This is achieved by throwing away each expert in S_t with a carefully chosen probability. Woodruff [Woo07] gave an elegant deterministic construction where the size of $S_t = O(\log t)$. We explain this in the appendix, and for the sake of clarity, do not give details in the algorithm description.

Advanced Follow-The-Leading-History

At round t, there is a set S_t of experts. Abusing notation, S_t will also denote the set of indices of the experts. At $t = 1$, $S_t = \{1\}$.

1. In round t, play $x_t = \sum_{j \in S_t} v_t^{(j)} x_t^{(j)}$ (or choose expert E^j with probability $v_t^{(j)}$).
2. Perform multiplicative update and addition to get vector $\overline{v}_{t+1}$.
3. Pruning step - Update S_t by removing some experts and adding $t+1$ to get S_{t+1}.
4. For all $i \in S_{t+1}$ -
$$v_{t+1}^{(i)} = \frac{\overline{v}_{t+1}^{(i)}}{\sum_{i \in S_{t+1}} \overline{v}_{t+1}^{(i)}}$$
The vector v_{t+1} (restricted to the experts in S_{t+1}) is a valid probability vector.

Now, when we wish to compute the regret incurred in a given interval, we can only compete with an expert that is present in working set throughout the interval. In such a situation, we get the same regret bounds as before. Our first step is to reprove Claim 3.5 in the new setting. We restate the claim for convenience.

Claim 4.3. *For any $i \in S_t$, the following are true -*

1. *For $i < t$, $\ln v_t^{(i)} \geq \ln \hat{v}_t^{(i)} - 2/t$*
2. *$\ln v_t^{(t)} \geq -2 \ln t$*

Proof. The claim is certainly true for $\overline{v}_t^{(i)}$. We note that $v_t^{(i)} \geq \overline{v}_t^{(i)}$ since $\sum_{i \in S_t} \overline{v}_t^{(i)} \leq 1$. □

The proof for the following lemma would be exactly the same of the proofs for Lemmas 3.3 and 3.7.

Lemma 4.4. *Consider some time interval $I = [r, s]$. Suppose that E^r was in the working set S_t, for all $t \in I$. Then the regret incurred in I is at most $R(T)$.*

Finally, we reach the main proof of this section. Given the properties of S_t, we can show that in any interval the regret incurred is small.

Lemma 4.5. *For interval I, the regret incurred by the AFLH for any interval I is at most $(R(T) + 1)(\log_2 |I| + 1)$.*

Proof. Let $|I| \in [2^k, 2^{k+1})$. We will prove by induction on k.

base case: For $k = 1$ the regret is bounded by $f_t(x_t) \leq R(T) \leq (R(T)+1)(\log_2 |I|+1)$.

induction step: By the properties of the S_t's, there is an expert E^i in the pool such that $i \in [r, (r + s)/2]$. This expert E^i entered the pool at time i and stayed throughout $[i, s]$. By Lemma 4.4, the algorithm incurs regret at most $R(T)$ in $[i, s]$.

The interval $[r, i - 1]$ has size in $[2^{k-1}, 2^k)$, and by induction the algorithm has regret of $(R(T) + 1)k$ on $[s, i]$. This gives a total of $(R(T) + 1)(k + 1)$ regret on I. □

The running time of AFLH is bounded by $|S_t|V$. Since $|S_t| = O(\log t)$, we can bound the running time by $O(V \log T)$. This fact, together with Lemma 4.5, complete the proof of Theorem 4.1.

References

[BCK03] A. Blum, S. Chawla, and A. Kalai. Static optimality and dynamic search optimality in lists and trees. *Algorithmica*, 36(3), 2003.

[BW03] Olivier Bousquet and Manfred K. Warmuth. Tracking a small set of experts by mixing past posteriors. *J. Mach. Learn. Res.*, 3:363–396, 2003.

[CBL06] Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[Cov91] T. Cover. Universal portfolios. *Math. Finance*, 1:1–19, 1991.

[FS97] Yoav Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

[GJKK] P. Gopalan, T.S. Jayram, R. Krauthgamer, and R. Kumar. Estimating the sortedness of a data stream. In *SODA 2007*.

[HKKA06] Elad Hazan, Adam Kalai, Satyen Kale, and Amit Agarwal. Logarithmic regret algorithms for online convex optimization. In *COLT*, pages 499–513, 2006.

[HW98] Mark Herbster and Manfred K. Warmuth. Tracking the best expert. *Mach. Learn.*, 32(2):151–178, 1998.

[KS07a] S.S. Kozat and A.C. Singer. Universal constant rebalanced portfolios with switching. In *tech report*, 2007.

[KS07b] S.S. Kozat and A.C. Singer. Universal piecewise constant and least squares prediction. In *tech report*, 2007.

[KV03] Adam Kalai and Santosh Vempala. Efficient algorithms for universal portfolios. *J. Mach. Learn. Res.*, 3:423–440, 2003.

[KV05] Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

[OC98] Erik Ordentlich and Thomas M. Cover. The cost of achieving the best portfolio in hindsight. *Mathematics of Operations Research*, 23(4):960–982, 1998.

[Sin] Yoram Singer. Switching portfolios. pages 488–495.

[ST85] D. Sleator and R. Tarjan. Self-adjusting binary search trees. *J. ACM*, 32, 1985.

[TW03] Eiji Takimoto and Manfred K. Warmuth. Path kernels and multiplicative updates. *J. Mach. Learn. Res.*, 4:773–818, 2003.

[Woo07] David Woodruff. personal communications. 2007.

[Zin03] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference (ICML)*, pages 928–936, 2003.

A Appendix

A.1 General convex loss functions

We give a proof for Lemma 3.7. Analogous to Lemma 3.4, the following gives bounds on the regret incurred in any round.

Lemma A.1. *Suppose that $\forall x \in K$ the value $f_t(x)$ is always positive and bounded by constant M.*

1. *For $i < t$ -*
$$f_t(x_t) \leq \frac{M(\ln \hat{v}_{t+1}^{(i)} - \ln \hat{v}_t^{(i)} + 2/t) + \eta_t M f_t(x_t^{(i)})}{1 - e^{-\eta_t M}}$$
2. $$f_t(x_t) \leq \frac{M(\ln \hat{v}_{t+1}^{(t)} + 2\ln t) + \eta_t M f_t(x_t^{(t)})}{1 - e^{-\eta_t M}}$$

Proof. We use relative entropy distances. For two n-dimensional vectors, u, v,

$$\Delta(u, v) = \sum_{i=1}^{n} u^{(i)} \ln \frac{u^{(i)}}{v^{(i)}}$$

Conventionally, $0 \ln 0 = 0$. We want to compare performance with respect to the ith expert, and therefore we set $\vec{u}$ to have 1 at the ith coordinate and zero elsewhere.

$$\begin{aligned}
\Delta(u, v_t) - \Delta(u, \hat{v}_{t+1}) &= -\ln v_t^{(i)} + \ln \hat{v}_{t+1}^{(i)} \\
&= -\ln v_t^{(i)} + \ln \frac{v_t^{(i)} e^{-\eta_t f_t(x_t^{(i)})}}{\sum_{j=1}^{t} v_t^{(j)} e^{-\eta_t f_t(x_t^{(j)})}} \\
&= -\eta_t f_t(x_t^{(i)}) - \ln(\sum_{j=1}^{t} v_t^{(j)} e^{-\eta_t f_t(x_t^{(j)})})
\end{aligned}$$

For $x < M$, we can use the approximation $e^{-\eta_t x} \leq 1 - (1 - e^{-\eta_t M})(x/M)$. This gives us -

$$\begin{aligned}
-\ln v_t^{(i)} + \ln \hat{v}_{t+1}^{(i)} &\geq -\eta_t f_t(x_t^{(i)}) - \ln\Big(\sum_{j=1}^{t} v_t^{(j)} \big(1 - (1 - e^{-\eta_t M})(f_t(x_t^{(i)})/M)\big)\Big) \\
&= -\eta_t f_t(x_t^{(i)}) - \ln\Big(\sum_{j=1}^{t} v_t^{(j)} - M^{-1}(1 - e^{-\eta_t M}) \sum_{j=1}^{t} v_t^{(j)} f_t(x_t^{(i)})\Big)
\end{aligned}$$

Noting that v_t is a probability vector and that $f_t(x_t) = \sum_{j=1}^{t} v_t^{(j)} f_t(x_t^{(i)})$ -

$$\begin{aligned}
-\ln v_t^{(i)} + \ln \hat{v}_{t+1}^{(i)} &\geq -\eta_t f_t(x_t^{(i)}) - \ln(1 - M^{-1}(1 - e^{-\eta_t M}) f_t(x_t)) \\
&\geq -\eta_t f_t(x_t^{(i)}) + M^{-1}(1 - e^{-\eta_t M}) f_t(x_t) \\
\implies f_t(x_t) &\leq \frac{M(\ln \hat{v}_{t+1}^{(i)} - \ln v_t^{(i)}) + \eta_t M f_t(x_t^{(i)})}{1 - e^{-\eta_t M}}
\end{aligned}$$

Application of Claim 3.5 completes the proof. □

Proof. (Lemma 3.7) For interval I, we sum up the bounds given in Lemma A.1. For $\alpha > 0$ sufficiently small, we set $\eta_t = -M^{-1} \ln(1 - \alpha)$.

$$\begin{aligned}
\sum_{t=r}^{s} f_t(x_t) &= f_r(x_r) + \sum_{t=r+1}^{s} f_t(x_t) \\
&\leq \frac{M(\ln \hat{v}_{r+1}^{(r)} + 2\ln r) - \ln(1-\alpha) f_r(x_r^{(r)})}{1 - e^{\ln(1-\alpha)}} + \\
&\quad \frac{\sum_{t=r+1}^{s} [M(\ln \hat{v}_{t+1}^{(r)} - \ln \hat{v}_t^{(r)} + 2/t) - \ln(1-\alpha) f_r(x_t^{(r)})]}{1 - e^{\ln(1-2\varepsilon)}} \\
&\leq \alpha^{-1}[M(\ln \hat{v}_{s+1}^{(r)} + \ln s)] - \alpha^{-1} \ln(1-\alpha) \sum_{t=r}^{s} f_r(x_t^{(r)}) \\
&\leq \alpha^{-1}(\alpha + \alpha^2) \sum_{t=r}^{s} f_r(x_t^{(r)}) + M\alpha^{-1} \ln s \\
&= (1+\alpha) \sum_{t=r}^{s} f_r(x_t^{(r)}) + M\alpha^{-1} \ln s
\end{aligned}$$

□

A.2 Variable learning rate

In the previous section we assumed prior knowledge of the number of game iterations T. We now show how to get $O(\sqrt{T} \ln T)$ Adaptive-Regret without knowing T in advance by changing the learning rate.

Lemma A.2. *For interval $I = [r, s]$, FLH achieves regret of $O(\sqrt{s} \ln s)$ without knowledge of the total time T.*

Lemma A.3. *1. For any $i < t$ - $f_t(x_t) - f_t(x_t^{(i)}) \leq \eta_t^{-1}(\ln \hat{v}_{t+1}^{(i)} - \ln \hat{v}_t^{(i)} + \eta_t^2 M^2 + 2/t)$*

2. $f_t(x_t) - f_t(x_t^{(t)}) \leq \eta_t^{-1}(\ln \hat{v}_{t+1}^{(t)} + \eta_t^2 M^2 + \ln t)$

The constant M is an upper bound on $(f_t(x))$.

Proof.

$$\begin{aligned}
\ln v_t^{(i)} - \ln \hat{v}_{t+1}^{(i)} &= \ln v_t^{(i)} - \ln \frac{v_t^{(i)} e^{-\eta_t f_t(x_t^{(i)})}}{\sum_{j=1}^{t} v_t^{(j)} e^{-\eta_t f_t(x_t^{(j)})}} \\
&= \eta_t f_t(x_t^{(i)}) + \ln(\sum_{j=1}^{t} v_t^{(j)} e^{-\eta_t f_t(x_t^{(j)})})
\end{aligned}$$

$$\begin{aligned}
\sum_{\ell=1}^{t} v_t^{(\ell)} \ln \frac{\hat{v}_{t+1}^{(\ell)}}{v_t^{(\ell)}} &= \sum_{\ell=1}^{t} v_t^{(\ell)} \ln \frac{e^{-\eta_t f_t(x_t^{(\ell)})}}{\sum_{j=1}^{t} v_t^{(j)} e^{-\eta_t f_t(x_t^{(j)})}} \\
&= -\eta_t \sum_{\ell=1}^{t} v_t^{(\ell)} f_t(x_t^{(\ell)}) - \ln(\sum_{j=1}^{t} v_t^{(j)} e^{-\eta_t f_t(x_t^{(j)})})
\end{aligned}$$

Using the convexity of f_t, and putting the above equations together, we get -

$$\begin{aligned} f_t(x_t) - f_t(x_t^{(i)}) &= f_t(\sum_{\ell=1}^{t} v_t^{(\ell)} x_t^{(\ell)}) - f_t(x_t^{(i)}) \\ &\leq \sum_{\ell=1}^{t} v_t^{(\ell)} f_t(x_t^{(\ell)}) - f_t(x_t^{(i)}) \\ &= \eta_t^{-1}\Big(\ln \hat{v}_{t+1}^{(i)} - \ln v_t^{(i)} - \sum_{\ell=1}^{t} v_t^{(\ell)} \ln \frac{\hat{v}_{t+1}^{(\ell)}}{v_t^{(\ell)}}\Big) \end{aligned}$$

$$\begin{aligned} -\sum_{\ell=1}^{t} v_t^{(\ell)} \ln \frac{\hat{v}_{t+1}^{(\ell)}}{v_t^{(\ell)}} &= \eta_t \sum_{\ell=1}^{t} v_t^{(\ell)} f_t(x_t^{(\ell)}) + \ln(\sum_{\ell=1}^{t} v_t^{(\ell)} e^{-\eta_t f_t(x_t^{(\ell)})}) \\ &= \eta_t \sum_{\ell=1}^{t} v_t^{(\ell)} f_t(x_t^{(\ell)}) + \ln\big(1 - (1 - \sum_{\ell=1}^{t} v_t^{(\ell)} e^{-\eta_t f_t(x_t^{(\ell)})})\big) \\ &\leq \eta_t \sum_{\ell=1}^{t} v_t^{(\ell)} f_t(x_t^{(\ell)}) - 1 + \sum_{\ell=1}^{t} v_t^{(\ell)} e^{-\eta_t f_t(x_t^{(\ell)})} \\ &\leq \eta_t \sum_{\ell=1}^{t} v_t^{(\ell)} f_t(x_t^{(\ell)}) - 1 + \sum_{\ell=1}^{t} v_t^{(\ell)} (1 - \eta_t f_t(x_t^{(\ell)}) + (\eta_t f_t(x_t^{(\ell)}))^2) \\ &= \eta_t^2 \sum_{\ell=1}^{t} v_t^{(\ell)} f_t(x_t^{(\ell)})^2 \end{aligned}$$

Putting the above together, we get -

$$f_t(x_t) - f_t(x_t^{(i)}) \leq \eta_t^{-1}\Big(\ln \hat{v}_{t+1}^{(i)} - \ln v_t^{(i)} + \eta_t^2 \sum_{\ell=1}^{t} v_t^{(\ell)} f_t(x_t^{(\ell)})^2\Big)$$

Using Claim 3.5, we have $\ln v_t^{(i)} \geq \ln \hat{v}_t^{(i)} - 2/t$ and that $\ln v_t^{(t)} \geq -t \ln t$. Substituting each of these above, we complete the proof. □

Proof. (Lemma A.2) As before, we sum up the regret bounds given by Lemma A.3 and set $\eta_t = 1/\sqrt{t}$.

$$\begin{aligned} \sum_{t=r}^{s}(f_t(x_t) - f_t(x_t^{(r)})) &= (f_p(x_p) - f_p(x_p^{(r)})) + \sum_{t=r+1}^{s} (f_t(x_t) - f_t(x_t^{(r)})) \\ &\leq \eta_p^{-1}(\ln \hat{v}_{r+1}^{(r)} + \eta_p^2 M^2 + 2\ln r) + \sum_{t=r+1}^{s} \eta_t^{-1}(\ln \hat{v}_{t+1}^{(r)} - \ln \hat{v}_t^{(r)} + \eta_t^2 M^2 + 2/t) \\ &= M^2 \eta_p + 2\eta_p^{-1} \ln r + \eta_q^{-1} \ln \hat{v}_{s+1}^{(r)} + \sum_{t=r+1}^{s} \ln \hat{v}_t^{(r)} (\eta_{t-1}^{-1} - \eta_t^{-1}) \\ &\quad + \sum_{t=r+1}^{s} M^2 \eta_t + \sum_{t=r+1}^{s} 2/(t\eta_t) \end{aligned}$$

Setting $\eta_t = 1/\sqrt{t}$ -

$$\sum_{t=r}^{s}(f_t(x_t) - f_t(x_t^{(r)})) \leq M^2/\sqrt{r} + 2\sqrt{r}\ln r - \sum_{t=r+1}^{s} \ln \hat{v}_t^{(r)}(\sqrt{t} - \sqrt{t-1}) + (M^2+2)(\sqrt{s} - \sqrt{r})$$

We now provide a lower bound for $\ln \hat{v}_t^{(r)}$, which will allow us to upper bound the regret. Since, in this case, $r < t$ -

$$\hat{v}_t^{(r)} = \frac{v_{t-1}^{(r)} e^{-\eta_{t-1} f_{t-1}(x_{t-1}^{(r)})}}{\sum_{j=1}^{t-1} v_{t-1}^{(j)} e^{-\eta_{t-1} f_{t-1}(x_t^{(j)})}}$$

Since $\forall t$, $\eta_t \leq 1$ and $f_t = \Theta(1)$, $e^{-\eta_{t-1} f_{t-1}(x_t^{(j)})} \in [c_1, c_2]$ (for some positive constants c_1, c_2). We also have that $v_{t-1}^{(r)} = 1/(t-1)$.

$$\hat{v}_t^{(r)} \geq \frac{c_1}{c_2(t-1)\sum_{j=1}^{t-1} v_{t-1}^{(j)}} = O(t^{-1})$$

Therefore, $-\ln \hat{v}_t^{(r)} \leq O(\ln s)$ and we get -

$$\sum_{t=r}^{s}(f_t(x_t) - f_t(x_t^{(r)})) \leq O(\sqrt{s}\ln s)$$

□

B The streaming problem

We now explain Woodruff's solution for maintaining the set $S_t \subseteq [1, n]$ in a streaming manner.

We specify the *lifetime* of integer i - if $i = r2^k$, where r is odd, then the lifetime of i is the interval $2^{k+2} + 1$. Suppose the lifetime of i is m. Then for any time $t \in [i, i+m]$, integer i is *alive* at t. The set S_t is simply the set of all integers that are alive at time t. Obviously, at time t, the only integer added to S_t is t - this immediately proves Property 3. We now prove the other properties -

Proof. (Property 1) We need to show that some integer in $[s, (s+t)/2]$ is alive at time t. This is trivially true when $t - s < 2$, since $t-1, t \in S_t$. Let 2^ℓ be the largest power of 2 such that $2^\ell \leq (t-s)/2$. There is some integer $x \in [s, (s+t)/2]$ such that $2^\ell | x$. The lifetime of x is larger than $2^\ell \times 2 + 1 > t - s$, and x is alive at t. □

Proof. (Property 2) For $0 \leq k \leq \lfloor \log t \rfloor$, let us count the number of integers of the form $r2^k$ (r odd) alive at t. The lifetime of these integers are $2^{k+2} + 1$. The only integers alive lie in the interval $[t - 2^{k+2} - 1, t]$. Since all of these integers of this form are separated by gaps of 2^k, there are at most a constant number of such integers alive at t. Totally, the size of S_t is $O(\log t)$. □

C Lazy version

Below we define a lazy version of FLH, called LFLH. We use the "coin-flipping" technique applied in [CBL06] for the "label-efficient prediction" problem. Essentially, we notice that the martingale arguments apply to any low regret algorithm, and even to low Adaptive-Regret algorithms, rather than the multiplicative weights algorithm which they analyze.

Lazy-Follow-The-Leading-History

1. Set $\tau = 1$.
2. In round t, flip a random ε-balanced coin and obtain the RV C_t.
3. If $C_t = 1$ do
   (a) set $g_\tau \triangleq \frac{1}{\varepsilon} f_t$
   (b) update $\tau \leftarrow \tau + 1$.
   (c) Apply FTLH to the function g_τ to obtain $x_\tau = x_t \leftarrow FTLH(g_\tau)$

   Else if $C_t = 0$, set $x_t \leftarrow x_{t-1}$

Theorem C.1. *Suppose that for all $x \in K$ and $t \in [T]$ $f_t(x) \in [0, M]$. Let $R(T)$ be an upper bound on the regret of the algorithm used to implement FLH over a history of length T. Then with high probability, for any $\varepsilon > 0$*

1. $$\textit{Adaptive-Regret}_T(LFLH) \leq R(T) + O(\frac{M\sqrt{T \log T}}{\varepsilon})$$
2. *Throughout the running time of LFLH, $x_t \neq x_{t-1}$ at most εT times.*

Lemma C.2. *Suppose that for all $x \in K$ and $t \in [T]$ $f_t(x) \in [0, M]$. Let $I = [r, s] \subseteq [T]$ be any time interval, and let $R(T)$ be an upper bound on the regret of the algorithm used to implement FLH over a history of length T. Then for any $\varepsilon > 0$ and $c > 10$, with probability at least $1 - \frac{1}{T^c}$ it holds that*

$$\textit{Regret}_I(LFLH) \leq R(T) + O(\frac{cM\sqrt{T} \log T}{\varepsilon})$$

Proof. Let $f_1, ..., f_T$ be the stream of online cost functions for LFLH. Recall that for each $t \in T$, C_t denotes the outcome of an independent binary coin flip which is one with probability ε. Let

$$\tilde{f}_t \triangleq \begin{cases} 0 & C_t = 0 \\ \frac{1}{\varepsilon} f_t & C_t = 1 \end{cases}$$

The regret of LFLH in a certain interval $I = [r, s] \subseteq [T]$ is

$$\text{Regret}_I = \sum_{t \in I} f_t(x_t) - f_t(x_I^*)$$

Where $x_I^* \triangleq \arg\min_{x \in K} \sum_{t \in I} f_t(x)$. This quantity is a random variable, since the strategies x_t played by LFLH are determined by random coin flips [4]. In order to bound this regret, we first relate it to another random variable, namely

$$Y_I \triangleq \sum_{t \in I} \tilde{f}_t(x_t) - \tilde{f}_t(x_I^*)$$

Observe, that Y_I is the regret of FLH on the interval I for the functions $\tilde{f}_t$. Since the bound on the magnitude of the functions $\tilde{f}_t$ is $\frac{M}{\varepsilon}$, we get by Theorem 3.6 -

$$Y_I \leq R(T) + O(\frac{M}{\varepsilon}\sqrt{T}\log T) \tag{2}$$

We proceed to prove that

$$\Pr[\text{Regret}_I - Y_I \geq 2\frac{\sqrt{cT\log T}}{\varepsilon}] \leq e^{-c\log T} \tag{3}$$

By equations (2) and (3) the Lemma is obtained.

Define the random variable Z_t as

$$Z_t \triangleq f_t(x_t) - f_t(x_I^*) - \tilde{f}_t(x_t) + \tilde{f}_t(x_I^*)$$

Notice that $\sum_{t \in I} Z_t = \text{Regret}_I - Y_I$ and $|Z_t| \leq \frac{4M}{\varepsilon}$. In addition, the sequence of random variables $Z_r, ..., Z_s$ is a martingale difference sequence [CBL06] with respect to the random coin flip variables $C_r, ..., C_s$ [5] since

$$\mathrm{E}[Z_t | C_r, ..., C_{t-1}] = 0$$

The reason is that given all previous coin flips, the point x_t is uniquely determined by the algorithm. The only random variable is $\tilde{f}_t$ and we know that its expectation is exactly f_t. We now use an extension to Azuma's inequality (see [CBL06]) which implies:

$$\Pr[\sum_{t \in I} Z_t \geq \delta|I|] \leq e^{-\frac{\delta^2|I|\varepsilon^2}{8M^2}}$$

Applying this to our case, with $\delta = \frac{8M\sqrt{c\log T}}{\varepsilon\sqrt{T}}$ we get

$$\Pr[\sum_{t \in I} Z_t \geq \delta|I|] \leq e^{-c\log T} \tag{4}$$

Hence with probability $\geq 1 - \frac{1}{T^c}$ we have

$$\sum_{t \in I} Z_t = \text{Regret}_I - Y_I \leq \delta|I| \leq \frac{cM\sqrt{T\log T}}{\varepsilon}$$

By equation (2) we get that with probability $\geq 1 - \frac{1}{T^c}$ we have

$$\text{Regret}_I \leq R(T) + O(\frac{M}{\varepsilon}\sqrt{T}\log T) + \frac{cM\sqrt{T\log T}}{\varepsilon}$$

□

Theorem C.1 follows easily from this lemma and the application of the union bound.

[4] We can henceforth assume that the previous coin tosses $C_1, ..., C_{r-1}$ are arbitrarily fixed, The following arguments hold for any such fixation of previous tosses.

[5] Recall our assumption that the bit flips $C_1, ..., C_{r-1}$ are fixed arbitrarily