Efficient learning algorithms for changing environments

Elad Hazan EHAZAN@CS.PRINCETON.EDU
C. Seshadhri CSESHA@CS.PRINCETON.EDU
IBM Almaden Research Center, 650 Harry Rd., San Jose, CA 95120 USA

Abstract

We study online learning in an oblivious changing environment. The standard measure of regret bounds the difference between the cost of the online learner and the best decision in hindsight. Hence, regret minimizing algorithms tend to converge to the static best optimum, clearly a suboptimal behavior in changing environments. On the other hand, various metrics proposed to strengthen regret and allow for more dynamic algorithms produce inefficient algorithms.

We propose a different performance metric which strengthens the standard metric of regret and measures performance with respect to a changing comparator. We then describe a series of data-streaming-based reductions which transform algorithms for minimizing (standard) regret into adaptive algorithms albeit incurring only poly-logarithmic computational overhead.

Using this reduction, we obtain *efficient* low adaptive-regret algorithms for the problem of online convex optimization. This can be applied to various learning scenarios, i.e. online portfolio selection, for which we describe experimental results showing the advantage of adaptivity.

Appearing in *Proceedings of the 26^{th} International Conference on Machine Learning*, Montreal, Canada, 2009.

1. Introduction

In online optimization the decision maker iteratively 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 iterations grows.

In scenarios in which the environment variables are sampled from some (unknown) distribution, regret minimization algorithms effectively "learn" the environment and approach the optimal strategy. However, if the underlying distribution changes, no such claim can be made.

When the environment undergoes many changes, regret may not be the best measure of performance. Various researchers noted the "static" behavior of regret-minimizing algorithms and generalized the concept of regret to allow for a changing prediction strategy (Herbster & Warmuth, 1998; Freund et al., 1997; Lehrer, 2003; Blum & Mansour, 2007). Although of great generality and importance to various scenarios, previous research fails to provide for *efficient* algorithms for continuous online optimization problems, in particular portfolio management.

In this paper we aim to strike a balance between efficiency and adaptivity: we define a new measure of regret called *adaptive regret* , which is less general than some previous approaches, but general enough to capture intuitive notions of adaptivity. We then then use sketching and data streaming techniques to design efficient learning algorithms.

Our main result is an efficiency preserving reduction which transforms any low regret algorithm into a low adaptive regret algorithm. Adaptive regret deals with the behavior of learning algorithms on contiguous intervals, which very intuitively captures how well it *tracks* the progress of the environment. We give several variants of online convex optimization algorithms whose cost on *any* contiguous sequence of iterations is within a small additive term with respect to the *local optimum* - i.e. the cost of the optimal decision on this particular sequence of iterations. In contrast, low regret guarantees small difference in cost only over the entire sequence of iterations and compared to the global optimum. In particular, we give the first efficient adaptive algorithm for online portfolio management.

1.1. Formal statement of results

In online convex optimization, in each round $t = 1, 2, ...$, the decision maker plays a point x_t from a convex domain $K \subseteq \mathbb{R}^n$. A convex 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*

$$\textit{Adaptive-Regret}_T(\mathcal{A}) \triangleq$$

$$\sup_{I=[r,s]\subseteq[T]} \left\{ \sum_{t=r}^{s} f_t(x_t) - \min_{x_I^* \in K} \sum_{t=r}^{s} f_t(x_I^*) \right\}$$

A crucial point in the above definition is that the comparison in cost is with respect to a *different* optimum for any interval, i.e. x_I^* can vary arbitrarily with I. Intuitively, this quantity measures on every interval of time how well we performed, compared to the optimum in hindsight *for that interval*. Ideally, we want the adaptive regret to be strongly sublinear.

Why is attaining low adaptive regret difficult? Here is a simple explanetory example in which current algorithms fail: Consider the simple example of square loss in one dimension. In each round, we play a point $x_t \in [-1, 1]$. The loss function f_t is one of the two functions $f_t(x) \in \{(x-1)^2, (x+1)^2\}$. For this scenario, the simple "follow-the-leader" algorithm, which plays the optimum decision so far - $x_{t+1} = t^{-1} \cdot \sum_{i=1}^{t} y_i$, where y_t is $\in \{\pm 1\}$ appropriately - is known to attain $O(\log T)$ regret (Cesa-Bianchi & Lugosi, 2006). Consider the case in which the function is $(x-1)^2$ for the first half of the game iterations, and then $(x+1)^2$ for the rest. The optimum is hindsight is the point 0, and that is exactly where all present algorithms that give $O(\log T)$ regret converge to. However, this behavior is disastrous in terms of adaptive regret- current low regret algorithms take too long (linear time) to converge to -1, and attain adaptive regret of $\Omega(T)$. The goal of standard Regret minimization is too encumbered with the past to be able to shift rapidly. In contrast, the following Theorem asserts an algorithm whose "shifting time" is poly-logarithmic.

Theorem 1.1. *For online convex optimization with exp-concave loss functions (i.e. the online portfolio selection problem), there exists an algorithm with Adaptive-Regret$_T$ $= O(\log^2 T)$* [1] *and running time poly$(n, \log T)$.*

A natural question that arises is whether the $O(\log^2 T)$ adaptive regret is tight. In the full version of this paper (see (Hazan & Seshadhri, 2007)) we show that in fact $O(\log T)$ adaptive regret is attainable and tight. However, the running time deteriorates to poly(n, T).

[1] We can also ensure that the "standard regret" remains $O(\log T)$.

1.2. Logarithmic vs. polynomial regret

The results stated above assume exp-concave loss functions (which apply to portfolio selection). Our reductions are general enough to apply to convex cost functions (i.e. linear costs) [2], however in this abstract we focus on the exp-concave case which allows for logarithmic, rather than the usual square-root, regret. The reason is that in this setting no polynomial time adaptive algorithms were known. In contrast, Zinkevich's algorithm for general convex functions (Zinkevich, 2003) does run in polynomial time and has adaptive guarantees. For the latter general convex case our improvement is in terms of efficiency only, although not as dramatic (for example, in online shortest paths, Zinkevich's algorithm would require to solve a convex program which takes time roughly $\tilde{O}(n^{3.5})$, vs. our implementation which would require $O(\log T)$ shortest path computations).

1.3. Relation to previous work

The notion of dealing with stronger performance notions than regret has been dealt with in (Herbster & Warmuth, 1998; Bousquet & Warmuth, 2003) on "tracking the best expert". Their focus 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. Freund et al. (1997) generalize this to a setting in which only a subset of the experts make predictions in different iterations. This is further generalized by Lehrer (2003) and Blum and Mansour (2007) to deal with more complicated situations where the total loss of an expert is computed by assigning a weight w_t to the loss of the expert in round t. The notion of adaptive regret is a (scaled) special case of these generalized regret definitions, in which temporal locality is given special importance.

Our setting differs from these expert settings in the following respects. We consider continuous decision sets rather than discrete. Although it is possible to discretize continuous sets and apply previous algorithms, such reductions are inefficient, resulting in exponential time algorithms. As for performance guarantees, the notion of adaptive regret generalizes (and is not equivalent to) regret versus the best *k-shifting* optimum: an algorithm with $O(R)$ adaptive regret obviously has $O(kR)$ regret against a k-shifting comparator. The converse is not necessarily true, since regret can be negative.

In independent work, Kozat and Singer (2007) attained related bounds of regret vs. a k-shifting strategy for the portfolio management problem. Our setting is more general, as it allows to tackle the general online convex optimization problem efficiently, and the techniques used are completely different.

2. Preliminaries

Various online problems can be modeled in the online convex optimization framework, as defined above. For example, the online portfolio selection problem (Cover, 1991) is modeled by taking the convex set to be the set of all distributions over n assets - i.e. the n dimensional simplex. The loss functions are taken to be $f_t(x) = -\log(x \cdot r_t)$, where r_t is the *return vector*, a non-negative vector which contains in each coordinate the ratio of closing to opening price for the corresponding asset.

We say that a loss function is α-exp-concave if the function $e^{-\alpha f(x)}$ is concave (i.e. the loss functions in online portfolio selection). For simplicity we henceforth assume that the cost functions are bounded over the decision set in absolute value by one (generalization is a simple matter of scaling).

We shall base our results on the following well-known fact from online learning theory:

Fact 2.1. *There exist algorithms for online convex optimization with α-exp-concave loss functions which attain regret $O(\frac{1}{\alpha}\log T)$ and run in time $O(n^3)$ (Hazan et al., 2006). Any online algorithm must suffer worst case regret of $Regret_T(OPT) = \Omega(\frac{1}{\alpha}\log T)$ (Cover, 1991).*

[2] In the full version of this manuscript, available at (Hazan & Seshadhri, 2007), this generalization will be detailed

3. The Algorithm

The basic idea of our algorithm is to reduce the continuous optimization problem at hand to the discrete realm of experts, which are themselves online optimization algorithms. We chose the "sub algorithms" such that at least one is guaranteed to have good performance at any game iteration.

To choose amongst the experts, we apply a twist on the well studied Multiplicative Weights method (the standard approach needs to be modified since our expert set keeps changing throughout the game and since we require additional adaptivity properties).

In order to improve efficiency, we prune the set of experts which are added online. We formally define the properties required of the ground set of experts, and the resulting recipe turns out to be an easily expressible abstract streaming problem. Incorporating the data streaming ideas yields an efficient algorithm.

3.1. Algorithm description

The basic algorithm, which we refer to as *Follow-the-Leading-History* (FLH), is detailed in the figure below. It uses many online algorithms, each attaining good regret for a different segment in history, and chooses 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 (1998). However, our set of experts continuously changes, as more algorithms are considered and others are discarded.

The experts that we use, denoted by $E^1, \cdots, E^T$ are low-regret algorithms that use different starting points in time to make predictions. The expert E^t will be a standard online algorithm that starts making predictions from round t and does *not* consider the history before time t. To maintain all of these experts simultaneously would be too time consuming. For the sake of efficiency, we maintain a *working set* of experts, S_t, that changes every round. At round t, the pertinent set of experts is $\{E^1, \cdots, E^t\}$ (abusing notation, we will refer to experts by their indices, so this set is just $[1, t]$). The set S_t will be a very sparse subset of $[1, t]$. After round t, S_t is updated to get S_{t+1}. This is done by adding $t + 1$ (the expert E^{t+1}) and removing some experts from S_t. Once removed, an expert cannot be added to the working set.

The problem of maintaining the set of active experts can be thought of as the following abstract data streaming problem. 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. A time t we create S_{t+1} from S_t: we are allowed to add to S_t only the integer $t + 1$, and remove any integer already in S_t. Our aim is to maintain a short "sketch" of the data seen so far.

We now describe the algorithm Follow-the-Leading-History. For the sake of clarity, we separately explain how the working set of experts is maintained.

Generation of working sets: We maintain the working sets using an algorithm due to Woodruff (2007) (there is another randomized solution to this streaming problem due to (Gopalan et al., 2007), which is simpler to apply but gives somewhat worse bounds). Any integer i can be uniquely written as $i = r2^k$ where r is odd. Let the *lifetime* of integer i be $2^{k+2} + 1$. 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.

Woodruff proved that the following properties are maintained by the scheme above. The first, and most important, property of the sets S_t essentially means that S_t is "well-spread" out in a logarithmic scale. This is depicted in Figure 1.

Figure 1. The set S_t

Property 3.1. *1. For every positive $s \leq t$, $[s,(s+t)/2] \cap S_t \neq \phi$.*

2. For all t, $|S_t| = O(\log T)$.

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

Algorithm 1 Follow-the-Leading-History (FLH)

Let $E^1, ..., E^T$ be online convex optimization algorithms. Let $S_t \in [t]$ be a set of experts, $S_1 = \{1\}$. Initialize $p_1^1 = 0$, for any t p_t is a distribution over S_t.

for $t = 1$ to T **do**

  Set $\forall j \in S_t$, $x_t^{(j)} \leftarrow E^j(f_{t-1})$ (the prediction of the j'th algorithm) and play $x_t = \sum_{j \in S_t} p_t^{(j)} x_t^{(j)}$.

  Multiplicative Update - After receiving f_t, set $\hat{p}_{t+1}^{(t+1)} = 0$ and perform update for $i \in S_t$ -

$$\hat{p}_{t+1}^{(i)} = \frac{p_t^{(i)} e^{-\alpha f_t(x_t^{(i)})}}{\sum_{j \in S_t} p_t^{(j)} e^{-\alpha f_t(x_t^{(j)})}}$$

  Addition step - Set $\overline{p}_{t+1}^{(t+1)}$ to $1/(t+1)$ and for $i \neq t+1$: $\overline{p}_{t+1}^{(i)} = (1-(t+1)^{-1}) p_{t+1}^{(i)}$

  Pruning step - Update S_t to the set S_{t+1}. For all $i \in S_{t+1}$:

$$p_{t+1}^{(i)} = \frac{\hat{p}_{t+1}^{(i)}}{\sum_{j \in S_{t+1}} \hat{p}_{t+1}^{(j)}}$$

end for

Note that the running time per round is bounded by the size of the S_t's times the running time of each expert. For efficiency, it is crucial that the S_t sets are very small. Yet for maintaining low adaptive regret, we will need the first and third properties, i.e. the well-spread out nature of S_t and the fact that every new integer "gets a chance" and is always added. We henceforth prove the following theorem:

Theorem 3.1. *If all loss functions are α-exp concave then the FLH algorithm attains adaptive regret of $O(\alpha^{-1} \log^2 T)$. The running time per iteration is $O(n^3 \log T)$.*

3.2. Proof of performance guarantees

The low adaptive regret guarantees of FLH are a consequence of the following two lemmas. The first lemma is obtain as a consequence of the "expert" algorithm applied to the set of experts S_t.

Lemma 3.1. *For any interval $I = [r,s]$ in time, suppose that E^r stays in the working set throughout I. Then, the algorithm FLH gives $O(\alpha^{-1}(\ln r + \ln |I|))$ regret with respect to the best optimum in hindsight for I.*

Before proving this lemma, let us state the following lemma in which the streaming algorithm comes into effect, and prove the main theorem.

Lemma 3.2. *For interval $I = [r,s]$, the regret incurred by the FLH for any interval I is at most $O(\frac{1}{\alpha} \log s \cdot \log |I| + 1)$.*

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

base case: For $k = 1$, the bound on the adaptive regret is an easy consequence of the fact that the cost functions are bounded in absolute value by one, hence $f_t(x_t) - f_t(x_t^*) \leq \log t \cdot \log 1 + 1 = 1$.

induction step: By the properties of the work sets $\{S_t\}$, there is an expert E^i in the working set S_s at time s, with $i \in [r,(r+s)/2]$, such that the following holds. This expert E^i entered the working set at time i and stayed throughout $[i,s]$. By Lemma 3.1, the algorithm incurs adaptive regret, and hence regret, of at most $c_1 \cdot \frac{1}{\alpha}(\log i + \log |I|)$ in $[i,s]$, for some $c_1 \geq 0$. Formally, since $s \geq |I|$,

$$\forall x^* \; . \; \sum_{t=i}^{s} f_t(x_t) - f_t(x^*) \leq \frac{c_1}{\alpha}(\log i + \log |I|) \leq \frac{2c_1}{\alpha} \log s$$

The interval $I_2 = [r, i-1]$ has size $|I_2| \in [2^{k-1}, 2^k)$ at most half of the entire interval I, and by induction the algorithm has regret of at most $\frac{c_2}{\alpha} \log |I_2| \cdot \log i \leq \frac{c_2}{\alpha} k \log s$ on this interval for some $c_2 > 0$, i.e.

$$\forall x^* \; . \; \sum_{t=r}^{i-1} f_t(x_t) - f_t(x^*) \leq \frac{c_2 k}{\alpha} \log s$$

Combining both previous equations,

$$\forall x^* \quad . \sum_{t=r}^{s} f_t(x_t) - f_t(x^*) \le \frac{c_2}{\alpha}(k + \frac{2c_1}{c_2}) \log s \le \frac{c_2(k+1)}{\alpha} \log s \le \frac{c_2}{\alpha} \log s \cdot \log |I|$$

Proving the induction hypothesis for a constant c_2 which satisfies $c_2 \ge 2c_1$. □

We can now deduce Theorem 1.1: By Fact 2.1, the running time of FLH is bounded by $O(|S_t|n^3)$. Since $|S_t| = O(\log t)$, we can bound the running time by $O(n^3 \log T)$. This fact, together with Lemma 3.2, completes the proof of Theorem 3.1. Note that by always keeping E^1 in the working set, we can ensure that the (standard) regret is bounded by $O(\log T)$.

We proceed to complete the missing step in the proof above, i.e. Lemma 3.1.

By assumption expert E^r gives $\frac{1}{\alpha} \log |I|$ 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 Lemma 3.1, it suffices to prove the following claim.

Claim 3.1. *For any $I = [r, s]$, suppose that E^r stays in the working set throughout I. 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 claim, which gives bounds on the regret in any round and then sum these bounds over all rounds.

Claim 3.2. *1. For $i \in S_t$,*

$$f_t(x_t) - f_t(x_t^{(i)}) \le \alpha^{-1}(\ln \hat{p}_{t+1}^{(i)} - \ln \hat{p}_t^{(i)} + 2/t)$$

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

Proof. Using the α-exp concavity of f_t -

$$\begin{aligned} e^{-\alpha f_t(x_t)} &= e^{-\alpha f_t(\sum_{j \in S_t} p_t^{(j)} x_t^{(j)})} \\ &\ge \sum_{j \in S_t} p_t^{(j)} e^{-\alpha f_t(x_t^{(j)})} \end{aligned}$$

Taking logarithm,

$$f_t(x_t) \le -\alpha^{-1} \ln \sum_{j \in S_t} p_t^{(j)} e^{-\alpha f_t(x_t^{(j)})}$$

Hence,

$$\begin{aligned} & f_t(x_t) - f_t(x_t^{(i)}) \\ \le\; & \alpha^{-1}(\ln e^{-\alpha f_t(x_t^{(i)})} - \ln \sum_{j \in S_t} p_t^{(j)} e^{-\alpha f_t(x_t^{(j)})}) \\ =\; & \alpha^{-1} \ln \frac{e^{-\alpha f_t(x_t^{(i)})}}{\sum_{j \in S_t} p_t^{(j)} e^{-\alpha f_t(x_t^{(j)})}} \\ =\; & \alpha^{-1} \ln (\frac{1}{p_t^{(i)}} \cdot \frac{p_t^{(i)} e^{-\alpha f_t(x_t^{(i)})}}{\sum_{j \in S_t} p_t^{(j)} e^{-\alpha f_t(x_t^{(j)})}}) \\ =\; & \alpha^{-1} \ln \frac{\hat{p}_{t+1}^{(i)}}{p_t^{(i)}} \qquad (1) \end{aligned}$$

The lemma is now obtained using the bounds of Claim 3.3 below.

□

Claim 3.3. *1. For $i \in S_t$, $\ln p_t^{(i)} \ge \ln \hat{p}_t^{(i)} - 2/t$*

2. $\ln p_t^{(t)} \ge -\ln t$

Proof. For $i \in S_t$, $p_t^{(i)} \ge \overline{p}_t^{(i)} = (1 - 1/t)\hat{p}_t^{(i)}$. Also, $p_t^{(t)} \ge \overline{p}_t^{(t)} = 1/t$. Taking the natural log of both these inequalities completes the proof. □

We are now ready to prove Claim 3.1.

Proof. (Claim 3.1) We are looking at regret in I with respect to an expert E^r. Since $r \in S_t$, for any $t \in I$, we can apply Claim 3.2.

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

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

4. Experimental evidence in support of adaptive regret

We implemented the Online Newton Step algorithm (ONS) of (Hazan et al., 2006; Agarwal et al., 2006), as well as the its adaptive version, as given in the above reductions. We used the exact same data set of (Agarwal et al., 2006) to repeat the same tests for these two algorithms. The tests are performed over a set of 50 random S&P 500 stocks and a period of 1000 trading days. Over this ground set of stocks, we chose $n \in [5, 10, 15, 20, 25]$ random stocks, and applied the two algorithms which were allowed to trade once every five trading days. This set of n random stocks is sampled one hundred times, and the final result is averaged.

The figure below depicts the performance of ONS and its adaptive version in terms of APY (annual percentage yield).

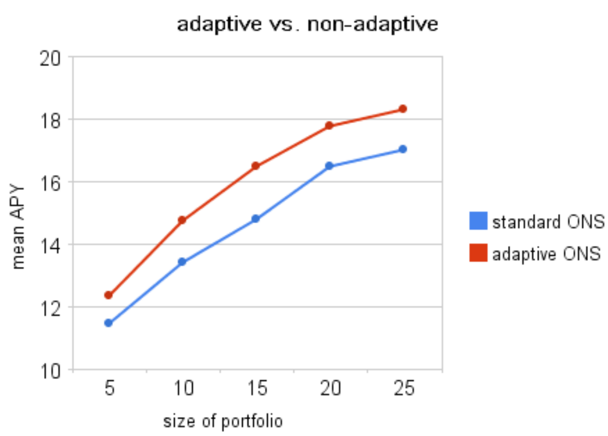

Figure 2. Standard ONS vs Adaptive ONS

As can be clearly seen, adaptivity gives a consistent one to two percent improvement in terms of APY, which amounts to roughly ten percent improvement in performance. Similar performance gains were observed in the other tests proposed by (Agarwal et al., 2006).

5. Conclusions and Future Work

In this paper we have investigated the notion of learning in a changing environment, and introduced online convex optimization algorithms which are adaptive in the sense that the converge to the local optimum of a contiguous time interval at almost the fastest possible rate. This is a generalization of previous adaptive algorithms, which were designed for the discrete setting, and allows us to tackle continuous problems such as portfolio management or problems with a large decision set i.e. online shortest paths.

As opposed to previous approaches, we base our reductions on streaming techniques and thus overcome the efficiency barrier which was prohibitive in previous approaches. However, the sketching/streaming techniques are tailored for the particular notion of adaptivity we consider, which is of time-locality, i.e. optimum for a contiguous interval in time. An interesting research direction is to generalize our result to even stronger notions of adaptivity, such as the "swap regret" notion of Blum and Mansour.

References

Agarwal, A., Hazan, E., Kale, S., & Schapire, R. (2006). Algorithms for portfolio management based on the newton method. *Proceedings of the 23rd International Conference on Machine Learning (ICML)* (pp. 9–16).

Blum, A., & Mansour, Y. (2007). From external to internal regret. *The Journal of Machine Learning Research*, *8*, 1307–1324.

Bousquet, O., & Warmuth, M. K. (2003). Tracking a small set of experts by mixing past posteriors. *J. Mach. Learn. Res.*, *3*, 363–396.

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

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

Freund, Y., Schapire, R. E., Singer, Y., & Warmuth., M. K. (1997). Using and combining predictors that specialize. *In Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing* (pp. 334–343).

Gopalan, P., Jayram, T. S., Krauthgamer, R., & Kumar, R. (2007). Estimating the sortedness of a data stream. *SODA* (pp. 318–327). SIAM.

Hazan, E., Kalai, A., Kale, S., & Agarwal, A. (2006). Logarithmic regret algorithms for online convex optimization. *Proceedings of 19th Annual Conference on Learning Theory, (COLT)* (pp. 499–513).

Hazan, E., & Seshadhri, C. (2007). Adaptive algorithms for online decision problems. *Electronic Colloquium on Computational Complexity (ECCC)*, *14*.

Herbster, M., & Warmuth, M. K. (1998). Tracking the best expert. *Mach. Learn.*, *32*, 151–178.

Kozat, S., & Singer, A. (2007). Universal constant rebalanced portfolios with switching. *IEEE International Conference on Acoustics, Speech and Signal Processing, (ICASSP)* (pp. 1129–1132).

Lehrer, E. (2003). A wide range no-regret theorem. *Games and Economic Behavior*, *42*, 101–115.

Woodruff, D. (2007). personal communications. .

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. *Proceedings of the Twentieth International Conference (ICML)* (pp. 928–936).

A. 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 $O(\log t)$. □