No-Regret Algorithms for Unconstrained Online Convex Optimization

Matthew Streeter
Duolingo, Inc.*
Pittsburgh, PA 15232
matt@duolingo.com

H. Brendan McMahan
Google, Inc.
Seattle, WA 98103
mcmahan@google.com

Abstract

Some of the most compelling applications of online convex optimization, including online prediction and classification, are unconstrained: the natural feasible set is $\mathbb{R}^n$. Existing algorithms fail to achieve sub-linear regret in this setting unless constraints on the comparator point $\mathring{x}$ are known in advance. We present algorithms that, without such prior knowledge, offer near-optimal regret bounds with respect to *any* choice of $\mathring{x}$. In particular, regret with respect to $\mathring{x} = 0$ is *constant*. We then prove lower bounds showing that our guarantees are near-optimal in this setting.

1 Introduction

Over the past several years, online convex optimization has emerged as a fundamental tool for solving problems in machine learning (see, e.g., [3, 12] for an introduction). The reduction from general online convex optimization to online linear optimization means that simple and efficient (in memory and time) algorithms can be used to tackle large-scale machine learning problems. The key theoretical techniques behind essentially all the algorithms in this field are the use of a fixed or increasing strongly convex regularizer (for gradient descent algorithms, this is equivalent to a fixed or decreasing learning rate sequence). In this paper, we show that a fundamentally different type of algorithm can offer significant advantages over these approaches. Our algorithms adjust their learning rates based not just on the number of rounds, but also based on the sum of gradients seen so far. This allows us to start with small learning rates, but effectively increase the learning rate if the problem instance warrants it.

This approach produces regret bounds of the form $\mathcal{O}\big(R\sqrt{T}\log((1+R)T)\big)$, where $R = \|\mathring{x}\|_2$ is the L_2 norm of an arbitrary comparator. Critically, our algorithms provide this guarantee simultaneously for *all* $\mathring{x} \in \mathbb{R}^n$, without any need to know R in advance. A consequence of this is that we can guarantee at most *constant* regret with respect to the origin, $\mathring{x} = 0$. This technique can be applied to any online convex optimization problem where a fixed feasible set is not an essential component of the problem. We discuss two applications of particular interest below:

Online Prediction Perhaps the single most important application of online convex optimization is the following prediction setting: the world presents an attribute vector $a_t \in \mathbb{R}^n$; the prediction algorithm produces a prediction $\sigma(a_t \cdot x_t)$, where $x_t \in \mathbb{R}^n$ represents the model parameters, and $\sigma : \mathbb{R} \to Y$ maps the linear prediction into the appropriate label space. Then, the adversary reveals the label $y_t \in Y$, and the prediction is penalized according to a loss function $\ell : Y \times Y \to \mathbb{R}$. For appropriately chosen σ and ℓ, this becomes a problem of online convex optimization against functions $f_t(x) = \ell(\sigma(a_t \cdot x), y_t)$. In this formulation, there are no inherent restrictions on the model coefficients $x \in \mathbb{R}^n$. The practitioner may have prior knowledge that "small" model vectors are more

*This work was performed while the author was at Google.

likely than large ones, but this is rarely best encoded as a feasible set $\mathcal{F}$, which says: "all $x_t \in \mathcal{F}$ are equally likely, and all other x_t are ruled out." A more general strategy is to introduce a fixed convex regularizer: L_1 and L_2^2 penalties are common, but domain-specific choices are also possible. While algorithms of this form have proved very effective at solving these problems, theoretical guarantees usually require fixing a feasible set of radius R, or at least an intelligent guess of the norm of an optimal comparator $\mathring{x}$.

The Unconstrained Experts Problem and Portfolio Management In the classic problem of predicting with expert advice (e.g., [3]), there are n experts, and on each round t the player selects an expert (say i), and obtains reward $g_{t,i}$ from a bounded interval (say $[-1, 1]$). Typically, one uses an algorithm that proposes a probability distribution p_t on experts, so the expected reward is $p_t \cdot g_t$.

Our algorithms apply to an unconstrained version of this problem: there are still n experts with payouts in $[-1, 1]$, but rather than selecting an individual expert, the player can place a "bet" of $x_{t,i}$ on each expert i, and then receives reward $\sum_i x_{t,i} g_{t,i} = x_t \cdot g_t$. The bets are unconstrained (betting a negative value corresponds to betting against the expert). In this setting, a natural goal is the following: place bets so as to achieve as much reward as possible, subject to the constraint that total losses are bounded by a constant (which can be set equal to some starting budget which is to be invested). Our algorithms can satisfy constraints of this form because regret with respect to $\mathring{x} = 0$ (which equals total loss) is bounded by a constant.

It is useful to contrast our results in this setting to previous applications of online convex optimization to portfolio management, for example [6] and [2]. By applying algorithms for exp-concave loss functions, they obtain log-wealth within $\mathcal{O}(\log(T))$ of the best constant rebalanced portfolio. However, this approach requires a "no-junk-bond" assumption: on each round, for each investment, you always retain at least an $\alpha > 0$ fraction of your initial investment. While this may be realistic (though not guaranteed!) for blue-chip stocks, it certainly is not for bets on derivatives that can lose all their value unless a particular event occurs (e.g., a stock price crosses some threshold). Our model allows us to handle such investments: if we play $x_i > 0$, an outcome of $g_i = -1$ corresponds exactly to losing 100% of that investment. Our results imply that if even one investment (out of exponentially many choices) has significant returns, we will increase our wealth exponentially.

Notation and Problem Statement For the algorithms considered in this paper, it will be more natural to consider reward-maximization rather than loss-minimization. Therefore, we consider online linear optimization where the goal is to maximize cumulative reward given adversarially selected linear reward functions $f_t(x) = g_t \cdot x$. On each round $t = 1 \dots T$, the algorithm selects a point $x_t \in \mathbb{R}^n$, receives reward $f_t(x_t) = g_t \cdot x_t$, and observes g_t. For simplicity, we assume $g_{t,i} \in [-1, 1]$, that is, $\|g_t\|_\infty \leq 1$. If the real problem is against convex loss functions $\ell_t(x)$, they can be converted to our framework by taking $g_t = -\nabla \ell_t(x_t)$ (see pseudo-code for REWARD-DOUBLING), using the standard reduction from online convex optimization to online linear optimization [13].

We use the compressed summation notation $g_{1:t} = \sum_{s=1}^{t} g_s$ for both vectors and scalars. We study the reward of our algorithms, and their regret against a fixed comparator $\mathring{x}$:

$$\text{Reward} \equiv \sum_{t=1}^{T} g_t \cdot x_t \qquad \text{and} \qquad \text{Regret}(\mathring{x}) \equiv g_{1:T} \cdot \mathring{x} - \sum_{t=1}^{T} g_t \cdot x_t.$$

Comparison of Regret Bounds The primary contribution of this paper is to establish matching upper and lower bounds for unconstrained online convex optimization problems, using algorithms that require no prior information about the comparator point $\mathring{x}$. Specifically, we present an algorithm that, for any $\mathring{x} \in \mathbb{R}^n$, guarantees $\text{Regret}(\mathring{x}) \leq \mathcal{O}\big(\|\mathring{x}\|_2 \sqrt{T} \log((1 + \|\mathring{x}\|_2)\sqrt{T})\big)$. To obtain this guarantee, we show that it is sufficient (and necessary) that reward is $\Omega(\exp(|g_{1:T}|/\sqrt{T}))$ (see Theorem 1). This shift of emphasis from regret-minimization to reward-maximization eliminates the quantification on $\mathring{x}$, and may be useful in other contexts.

Table 1 compares the bounds for REWARD-DOUBLING (this paper) to those of two previous algorithms: online gradient descent [13] and projected exponentiated gradient descent [8, 12]. For each

Our bounds are not directly comparable to the bounds cited above: a $\mathcal{O}(\log(T))$ regret bound on log-wealth implies wealth at least $\mathcal{O}\big(\text{OPT}/T\big)$, whereas we guarantee wealth like $\mathcal{O}\big(\text{OPT'} - \sqrt{T}\big)$. But more importantly, the comparison classes are different.

Assuming $\|g_t\|_2 \leq 1$:

	$\mathring{x} = 0$	$\|\mathring{x}\|_2 \leq R$	Arbitrary $\mathring{x}$
Gradient Descent, $\eta = \frac{R}{\sqrt{T}}$	$R\sqrt{T}$	$R\sqrt{T}$	$\|\mathring{x}\|_2 T$
REWARD-DOUBLING	ϵ	$R\sqrt{T}\log\left(\frac{n(1+R)T}{\epsilon}\right)$	$\|\mathring{x}\|_2\sqrt{T}\log\left(\frac{n(1+\|\mathring{x}\|_2)T}{\epsilon}\right)$

Assuming $\|g_t\|_\infty \leq 1$:

	$\mathring{x} = 0$	$\|\mathring{x}\|_1 \leq R$	Arbitrary $\mathring{x}$
Exponentiated G.D.	$R\sqrt{T\log n}$	$R\sqrt{T\log n}$	$\|\mathring{x}\|_1 T$
REWARD-DOUBLING	ϵ	$R\sqrt{T}\log\left(\frac{n(1+R)T}{\epsilon}\right)$	$\|\mathring{x}\|_1\sqrt{T}\log\left(\frac{n(1+\|\mathring{x}\|_1)\sqrt{T}}{\epsilon}\right)$

Table 1: Worst-case regret bounds for various algorithms (up to constant factors). Exponentiated G.D. uses feasible set $\{x : \|x\|_1 \leq R\}$, and REWARD-DOUBLING uses $\epsilon_i = \frac{\epsilon}{n}$ in both cases.

algorithm, we consider a fixed choice of parameter settings and then look at how regret changes as we vary the comparator point $\mathring{x}$.

Gradient descent is minimax-optimal [1] when the comparator point is contained in a hypershere whose radius is known in advance ($\|\mathring{x}\|_2 \leq R$) and gradients are sparse ($\|g_t\|_2 \leq 1$, top table). Exponentiated gradient descent excels when gradients are dense ($\|g_t\|_\infty \leq 1$, bottom table) but the comparator point is sparse ($\|\mathring{x}\|_1 \leq R$ for R known in advance). In both these cases, the bounds for REWARD-DOUBLING match those of the previous algorithms up to logarithmic factors, even when they are tuned optimally with knowledge of R.

The advantage of REWARD-DOUBLING shows up when the guess of R used to tune the competing algorithms turns out to be wrong. When $\mathring{x} = 0$, REWARD-DOUBLING offers constant regret compared to $\Omega(\sqrt{T})$ for the other algorithms. When $\mathring{x}$ can be arbitrary, only REWARD-DOUBLING offers sub-linear regret (and in fact its regret bound is optimal, as shown in Theorem 8).

In order to guarantee constant origin-regret, REWARD-DOUBLING frequently "jumps" back to playing the origin, which may be undesirable in some applications. In Section 4 we introduce SMOOTH-REWARD-DOUBLING, which achieves similar guarantees without resetting to the origin.

Related Work Our work is related, at least in spirit, to the use of a momentum term in stochastic gradient descent for back propagation in neural networks [7, 11, 9]. These results are similar in motivation in that they effectively yield a larger learning rate when many recent gradients point in the same direction.

In Follow-The-Regularized-Leader terms, the exponentiated gradient descent algorithm with unnormalized weights of Kivinen and Warmuth [8] plays $x_{t+1} = \arg\min_{x \in \mathbb{R}^n_+} g_{1:t} \cdot x + \frac{1}{\eta}(x \log x - x)$, which has closed-form solution $x_{t+1} = \exp(-\eta g_{1:t})$. Like our algorithm, this algorithm moves away from the origin exponentially fast, but unlike our algorithm it can incur arbitrarily large regret with respect to $\mathring{x} = 0$. Theorem 9 shows that no algorithm of this form can provide bounds like the ones proved in this paper.

Hazan and Kale [5] give regret bounds in terms of the variance of the g_t. Letting $G = |g_{1:t}|$ and $H = \sum_{t=1}^T g_t^2$, they prove regret bounds of the form $\mathcal{O}(\sqrt{V})$ where $V = H - G^2/T$. This result has some similarity to our work in that $G/\sqrt{T} = \sqrt{H - V}$, and so if we hold H constant, then when V is low, the critical ratio $G/\sqrt{T}$ that appears in our bounds is large. However, they consider the case of a known feasible set, and their algorithm (gradient descent with a constant learning rate) cannot obtain bounds of the form we prove.

2 Reward and Regret

In this section we present a general result that converts lower bounds on reward into upper bounds on regret, for one-dimensional online linear optimization. In the unconstrained setting, this result will be sufficient to provide guarantees for general n-dimensional online convex optimization.

Theorem 1. *Consider an algorithm for one-dimensional online linear optimization that, when run on a sequence of gradients $g_1, g_2, \ldots, g_T$, with $g_t \in [-1, 1]$ for all t, guarantees*

$$\text{Reward} \geq \kappa \exp\left(\gamma |g_{1:T}|\right) - \epsilon, \tag{1}$$

where $\gamma, \kappa > 0$ and $\epsilon \geq 0$ are constants. Then, against any comparator $\mathring{x} \in [-R, R]$, we have

$$\text{Regret}(\mathring{x}) \leq \frac{R}{\gamma}\left(\log\left(\frac{R}{\kappa\gamma}\right) - 1\right) + \epsilon, \tag{2}$$

letting $0 \log 0 = 0$ when $R = 0$. Further, any algorithm with the regret guarantee of Eq. (2) *must guarantee the reward of Eq.* (1)*.*

We give a proof of this theorem in the appendix. The duality between reward and regret can also be seen as a consequence of the fact that $\exp(x)$ and $y \log y - y$ are convex conjugates. The γ term typically contains a dependence on T like $1/\sqrt{T}$. This bound holds for all R, and so for some small R the $\log$ term becomes negative; however, for real algorithms the ϵ term will ensure the regret bound remains positive. The minus one can of course be dropped to simplify the bound further.

3 Gradient Descent with Increasing Learning Rates

In this section we show that allowing the learning rate of gradient descent to sometimes increase leads to novel theoretical guarantees.

To build intuition, consider online linear optimization in one dimension, with gradients $g_1, g_2, \ldots, g_T$, all in $[-1, 1]$. In this setting, the reward of unconstrained gradient descent has a simple closed form:

Lemma 2. *Consider unconstrained gradient descent in one dimension, with learning rate η. On round t, this algorithm plays the point $x_t = \eta g_{1:t-1}$. Letting $G = |g_{1:t}|$ and $H = \sum_{t=1}^{T} g_t^2$, the cumulative reward of the algorithm is exactly*

$$\text{Reward} = \frac{\eta}{2}\left(G^2 - H\right).$$

We give a simple direct proof in Appendix A. Perhaps surprisingly, this result implies that the reward is totally independent of the order of the linear functions selected by the adversary. Examining the expression in Lemma 2, we see that the optimal choice of learning rate η depends fundamentally on two quantities: the absolute value of the sum of gradients (G), and the sum of the squared gradients (H). If $G^2 > H$, we would like to use as large a learning rate as possible in order to maximize reward. In contrast, if $G^2 < H$, the algorithm will obtain negative reward, and the best it can do is to cut its losses by setting η as small as possible.

One of the motivations for this work is the observation that the state-of-the-art online gradient descent algorithms adjust their learning rates based *only* on the observed value of H (or its upper bound T); for example [4, 10]. We would like to increase reward by also accounting for G. But unlike H, which is monotonically increasing with time, G can both increase and decrease. **This makes simple guess-and-doubling tricks fail** when applied to G, and necessitates a more careful approach.

3.1 Analysis in One Dimension

In this section we analyze algorithm REWARD-DOUBLING-1D (Algorithm 1), which consists of a series of epochs. We suppose for the moment that an upper bound $\bar{H}$ on $H = \sum_{t=1}^{T} g_t^2$ is known in advance. In the first epoch, we run gradient descent with a small initial learning rate $\eta = \eta_1$. Whenever the total reward accumulated in the current epoch reaches $\eta\bar{H}$, we double η and start a new epoch (returning to the origin and forgetting all previous gradients except the most recent one).

Lemma 3. *Applied to a sequence of gradients $g_1, g_2, \ldots, g_T$, all in $[-1, 1]$, where $H = \sum_{t=1}^{T} g_t^2 \leq \bar{H}$,* REWARD-DOUBLING-1D *obtains reward satisfying*

$$\text{Reward} = \sum_{t=1}^{T} x_t g_t \geq \frac{1}{4}\eta_1 \bar{H} \exp\left(a\frac{|g_{1:T}|}{\sqrt{\bar{H}}}\right) - \eta_1 \bar{H}, \tag{3}$$

for $a = \log(2)/\sqrt{3}$.

Algorithm 1 REWARD-DOUBLING-1D

Parameters: initial learning rate η_1, upper bound $\bar{H} \geq \sum_{t=1}^{T} g_t^2$.
Initialize $x_1 \leftarrow 0$, $i \leftarrow 1$, and $Q_1 \leftarrow 0$.
for $t = 1, 2, \ldots, T$ **do**
  Play x_t, and receive reward $x_t g_t$.
  $Q_i \leftarrow Q_i + x_t g_t$.
  if $Q_i < \eta_i \bar{H}$ **then**
    $x_{t+1} \leftarrow x_t + \eta_i g_t$.
  else
    $i \leftarrow i + 1$.
    $\eta_i \leftarrow 2\eta_{i-1}$; $Q_i \leftarrow 0$.
    $x_{t+1} \leftarrow 0 + \eta_i g_t$.

Algorithm 2 REWARD-DOUBLING

Parameters: maximum origin-regret ϵ_i for $1 \leq i \leq n$.
for $i = 1, 2, \ldots, n$ **do**
  Let A_i be a copy of algorithm REWARD-DOUBLING-1D-GUESS (see Theorem 4), with parameter ϵ_i.
for $t = 1, 2, \ldots, T$ **do**
  Play x_t, with $x_{t,i}$ selected by A_i.
  Receive gradient vector $g_t = -\nabla f_t(x_t)$.
  for $i = 1, 2, \ldots, n$ **do**
    Feed back $g_{t,i}$ to A_i.

Proof. Suppose round T occurs during the k'th epoch. Because epoch i can only come to an end if $Q_i \geq \eta_i \bar{H}$, where $\eta_i = 2^{i-1}\eta_1$, we have

$$\text{Reward} = \sum_{i=1}^{k} Q_i \geq \left(\sum_{i=1}^{k-1} 2^{i-1} \eta_1 \bar{H} \right) + Q_k = \left(2^{k-1} - 1\right) \eta_1 \bar{H} + Q_k \,. \tag{4}$$

We now lower bound Q_k. For $i = 1, \ldots, k$ let t_i denote the round on which Q_i is initialized to 0, with $t_1 \equiv 1$, and define $t_{k+1} \equiv T$. By construction, Q_i is the total reward of a gradient descent algorithm that is active on rounds t_i through t_{i+1} inclusive, and that uses learning rate η_i (note that on round t_i, this algorithm gets 0 reward and we initialize Q_i to 0 on that round). Thus, by Lemma 2, we have that for any i,

$$Q_i = \frac{\eta_i}{2} \left((g_{t_i:t_{i+1}})^2 - \sum_{s=t_i}^{t_{i+1}} g_s^2 \right) \geq -\frac{\eta_i}{2} \bar{H} \,.$$

Applying this bound to epoch k, we have $Q_k \geq -\frac{1}{2}\eta_k \bar{H} = -2^{k-2}\eta_1 \bar{H}$. Substituting into (4) gives

$$\text{Reward} \geq \eta_1 \bar{H}(2^{k-1} - 1 - 2^{k-2}) = \eta_1 \bar{H}(2^{k-2} - 1) \,. \tag{5}$$

We now show that $k \geq \frac{|g_{1:T}|}{\sqrt{3\bar{H}}}$. At the end of round $t_{i+1} - 1$, we must have had $Q_i < \eta_i \bar{H}$ (otherwise epoch $i + 1$ would have begun earlier). Thus, again using Lemma 2,

$$\frac{\eta_i}{2} \left((g_{t_i:t_{i+1}-1})^2 - \bar{H} \right) \leq \eta_i \bar{H}$$

so $|g_{t_i:t_{i+1}-1}| \leq \sqrt{3\bar{H}}$. Thus,

$$|g_{1:T}| \leq \sum_{i=1}^{k} |g_{t_i:t_{i+1}-1}| \leq k\sqrt{3\bar{H}} \,.$$

Rearranging gives $k \geq \frac{|g_{1:T}|}{\sqrt{3\bar{H}}}$, and combining with Eq. (5) proves the lemma. □

We can now apply Theorem 1 to the reward (given by Eq. (3)) of REWARD-DOUBLING-1D to show

$$\text{Regret}(\mathring{x}) \leq bR\sqrt{\bar{H}} \left(\log \left(\frac{4Rb\sqrt{\bar{H}}}{\eta_1} \right) - 1 \right) + \eta_1 \bar{H} \tag{6}$$

for any $\mathring{x} \in [-R, R]$, where $b = a^{-1} = \sqrt{3}/\log(2) < 2.5$. When the feasible set is also fixed in advance, online gradient descent with a fixed learning obtains a regret bound of $\mathcal{O}(R\sqrt{T})$. Suppose we use the estimate $\bar{H} = T$. By choosing $\eta_1 = \frac{1}{T}$, we guarantee constant regret against the origin, $\mathring{x} = 0$ (equivalently, constant total loss). Further, for *any* feasible set of radius R, we still have

worst-case regret of at most $\mathcal{O}(R\sqrt{T}\log((1+R)T))$, which is only modestly worse than that of gradient descent with the optimal R known in advance.

The need for an upper bound $\bar{H}$ can be removed using a standard guess-and-doubling approach, at the cost of a constant factor increase in regret (see appendix for proof).

Theorem 4. *Consider algorithm* REWARD-DOUBLING-1D-GUESS*, which behaves as follows. On each era i, the algorithm runs* REWARD-DOUBLING-1D *with an upper bound of $\bar{H}_i = 2^{i-1}$, and initial learning rate $\eta_1^i = \epsilon 2^{-2i}$. An era ends when $\bar{H}_i$ is no longer an upper bound on the sum of squared gradients seen during that era. Letting $c = \frac{\sqrt{2}}{\sqrt{2}-1}$, this algorithm has regret at most*

$$\text{Regret} \leq cR\sqrt{H+1}\left(\log\left(\frac{R}{\epsilon}(2H+2)^{5/2}\right)-1\right)+\epsilon.$$

3.2 Extension to n dimensions

To extend our results to general online convex optimization, it is sufficient to run a separate copy of REWARD-DOUBLING-1D-GUESS for each coordinate, as is done in REWARD-DOUBLING (Algorithm 2). The key to the analysis of this algorithm is that overall regret is simply the sum of regret on n one-dimensional subproblems which can be analyzed independently.

Theorem 5. *Given a sequence of convex loss functions $f_1, f_2, \ldots, f_T$ from $\mathbb{R}^n$ to $\mathbb{R}$,* REWARD-DOUBLING *with $\epsilon_i = \frac{\epsilon}{n}$ has regret bounded by*

$$\begin{aligned}\text{Regret}(\mathring{x}) &\leq \epsilon + c\sum_{i=1}^{n}|\mathring{x}_i|\sqrt{H_i+1}\left(\log\left(\frac{n}{\epsilon}|\mathring{x}_i|(2H_i+2)^{5/2}\right)-1\right)\\ &\leq \epsilon + c\|\mathring{x}\|_2\sqrt{H+n}\left(\log\left(\frac{n}{\epsilon}\|\mathring{x}\|_2^2(2H+2)^{5/2}\right)-1\right)\end{aligned}$$

for $c = \frac{\sqrt{2}}{\sqrt{2}-1}$, where $H_i = \sum_{t=1}^{T} g_{t,i}^2$ and $H = \sum_{t=1}^{T}\|g_t\|_2^2$.

Proof. Fix a comparator $\mathring{x}$. For any coordinate i, define

$$\text{Regret}_i = \sum_{t=1}^{T}\mathring{x}_i g_{t,i} - \sum_{t=1}^{T} x_{t,i}g_{t,i}\,.$$

Observe that

$$\sum_{i=1}^{n}\text{Regret}_i = \sum_{t=1}^{T}\mathring{x}\cdot g_t - \sum_{t=1}^{T} x_t\cdot g_t = \text{Regret}(\mathring{x})\,.$$

Furthermore, Regret_i is simply the regret of REWARD-DOUBLING-1D-GUESS on the gradient sequence $g_{1,i}, g_{2,i}, \ldots, g_{T,i}$. Applying the bound of Theorem 4 to each Regret_i term completes the proof of the first inequality. For the second inequality, let $\vec{H}$ be a vector whose i^{th} component is $\sqrt{H_i+1}$, and let $\vec{x}\in\mathbb{R}^n$ where $\vec{x}_i = |\mathring{x}_i|$. Using the Cauchy-Schwarz inequality, we have

$$\sum_{i=1}^{n}|\mathring{x}_i|\sqrt{H_i+1} = \vec{x}\cdot\vec{H} \leq \|\mathring{x}\|_2\,\|\vec{H}\|_2 = \|\mathring{x}\|_2\sqrt{H+n}\,.$$

This, together with the fact that $\log(|\mathring{x}_i|(2H_i+2)^{5/2}) \leq \log(\|\mathring{x}\|_2^2(2H+2)^{5/2})$, suffices to prove second inequality. □

In some applications, n is not known in advance. In this case, we can set $\epsilon_i = \frac{\epsilon}{i^2}$ for the ith coordinate we encounter, and get the same bound up to constant factors.

4 An Epoch-Free Algorithm

In this section we analyze SMOOTH-REWARD-DOUBLING, a simple algorithm that achieves bounds comparable to those of Theorem 4, without guessing-and-doubling. We consider only the 1-d problem, as the technique of Theorem 5 can be applied to extend to n dimensions. Given a parameter

$\eta > 0$, we achieve

$$\text{Regret} \le R\sqrt{T}\left(\log\left(\frac{RT^{3/2}}{\eta}\right) - 1\right) + 1.76\eta, \tag{7}$$

for all T and R, which is better (by constant factors) than Theorem 4 when $g_t \in \{-1, 1\}$ (which implies $T = H$). The bound can be worse on a problems where $H < T$.

The idea of the algorithm is to maintain the invariant that our cumulative reward, as a function of $g_{1:t}$ and t, satisfies $\text{Reward} \ge N(g_{1:t}, t)$, for some fixed function N. Because reward changes by $g_t x_t$ on round t, it suffices to guarantee that for any $g \in [-1, 1]$,

$$N(g_{1:t}, t) + g x_{t+1} \ge N(g_{1:t} + g, t + 1) \tag{8}$$

where x_{t+1} is the point the algorithm plays on round $t + 1$, and we assume $N(0, 1) = 0$.

This inequality is approximately satisfied (for small g) if we choose

$$x_{t+1} = \frac{\partial N(g_{1:t} + g, t)}{\partial g} \approx \frac{N(g_{1:t} + g, t) - N(g_{1:t}, t)}{g} \approx \frac{N(g_{1:t} + g, t + 1) - N(g_{1:t}, t)}{g}.$$

This suggests that if we want to maintain reward at least $N(g_{1:t}, t) = \frac{1}{t}(\exp(|g_{1:t}|/\sqrt{t}) - 1)$, we should set $x_{t+1} \approx \text{sign}(g_{1:t}) t^{-3/2} \exp\left(\frac{|g_{1:t}|}{\sqrt{t}}\right)$. The following theorem (proved in the appendix) provides an inductive analysis of an algorithm of this form.

Theorem 6. *Fix a sequence of reward functions $f_t(x) = g_t x$ with $g_t \in [-1, 1]$, and let $G_t = |g_{1:t}|$. We consider* SMOOTH-REWARD-DOUBLING*, which plays 0 on round 1 and whenever $G_t = 0$; otherwise, it plays*

$$x_{t+1} = \eta\, \text{sign}(g_{1:t}) B(G_t, t + 5) \tag{9}$$

with $\eta > 0$ a learning-rate parameter and

$$B(G, t) = \frac{1}{t^{3/2}} \exp\left(\frac{G}{\sqrt{t}}\right). \tag{10}$$

Then, at the end of each round t, this algorithm has

$$\text{Reward}(t) \ge \eta \frac{1}{t + 5} \exp\left(\frac{G_t}{\sqrt{t + 5}}\right) - 1.76\eta.$$

Two main technical challenges arise in the proof: first, we prove a result like Eq. (8) for $N(g_{1:t}, t) = (1/t) \exp\left(|g_{1:t}|/\sqrt{t}\right)$. However, this Lemma only holds for $t \ge 6$ and when the sign of $g_{1:t}$ doesn't change. We account for this by showing that a small modification to N (costing only a constant over all rounds) suffices.

By running this algorithm independently for each coordinate using an appropriate choice of η, one can obtain a guarantee similar to that of Theorem 5.

5 Lower Bounds

As with our previous results, it is sufficient to show a lower bound in one dimension, as it can then be replicated independently in each coordinate to obtain an n dimensional bound. Note that our lower bound contains the factor $\log(|\mathring{x}|\sqrt{T})$, which can be negative when $\mathring{x}$ is small relative to T, hence it is important to hold $\mathring{x}$ fixed and consider the behavior as $T \to \infty$. Here we give only a proof sketch; see Appendix A for the full proof.

Theorem 7. *Consider the problem of unconstrained online linear optimization in one dimension, and an online algorithm that guarantees origin-regret at most ϵ. Then, for any fixed comparator $\mathring{x}$, and any integer T_0, there exists a gradient sequence $\{g_t\} \in [-1, 1]^T$ of length $T \ge T_0$ for which the algorithm's regret satisfies*

$$\text{Regret}(\mathring{x}) \ge 0.336|\mathring{x}|\sqrt{T \log\left(\frac{|\mathring{x}|\sqrt{T}}{\epsilon}\right)}.$$

Proof. (Sketch) Assume without loss of generality that $\mathring{x} > 0$. Let Q be the algorithm's reward when each g_t is drawn independently uniformly from $\{-1, 1\}$. We have $\mathrm{E}[Q] = 0$, and because the algorithm guarantees origin-regret at most ϵ, we have $Q \geq -\epsilon$ with probability 1. Letting $G = g_{1:T}$, it follows that for any threshold $Z = Z(T)$,

$$\begin{aligned} 0 &= \mathrm{E}[Q] \\ &= \mathrm{E}[Q|G < Z] \cdot \Pr[G < Z] + \mathrm{E}[Q|G \geq Z] \cdot \Pr[G \geq Z] \\ &\geq -\epsilon \Pr[G < Z] + \mathrm{E}[Q|G \geq Z] \cdot \Pr[G \geq Z] \\ &> -\epsilon + \mathrm{E}[Q|G \geq Z] \cdot \Pr[G \geq Z] \,. \end{aligned}$$

Equivalently,

$$\mathrm{E}[Q|G \geq Z] < \frac{\epsilon}{\Pr[G \geq Z]} \,.$$

We choose $Z(T) = \sqrt{kT}$, where $k = \left\lfloor \log(\frac{R\sqrt{T}}{\epsilon}) / \log(p^{-1}) \right\rfloor$. Here $R = |\mathring{x}|$ and $p > 0$ is a constant chosen using binomial distribution lower bounds so that $\Pr[G \geq Z] \geq p^k$. This implies

$$\mathrm{E}[Q|G \geq Z] < \epsilon p^{-k} = \epsilon \exp\left(k \log p^{-1}\right) \leq R\sqrt{T} \,.$$

This implies there exists a sequence with $G \geq Z$ and $Q < R\sqrt{T}$. On this sequence, regret is at least $G\mathring{x} - Q \geq R\sqrt{kT} - R\sqrt{T} = \Omega(R\sqrt{kT})$. □

Theorem 8. *Consider the problem of unconstrained online linear optimization in $\mathbb{R}^n$, and consider an online algorithm that guarantees origin-regret at most ϵ. For any radius R, and any T_0, there exists a gradient sequence gradient sequence $\{g_t\} \in ([-1, 1]^n)^T$ of length $T \geq T_0$, and a comparator $\mathring{x}$ with $\|\mathring{x}\|_1 = R$, for which the algorithm's regret satisfies*

$$\mathrm{Regret}(\mathring{x}) \geq 0.336 \sum_{i=1}^{n} |\mathring{x}_i| \sqrt{T \log\left(\frac{|\mathring{x}_i|\sqrt{T}}{\epsilon}\right)} \,.$$

Proof. For each coordinate i, Theorem 7 implies that there exists a $T \geq T_0$ and a sequence of gradients $g_{t,i}$ such that

$$\sum_{t=1}^{T} \mathring{x}_i g_{t,i} - \sum_{t=1}^{T} x_{t,i} g_{t,i} \geq 0.336 |\mathring{x}_i| \sqrt{T \log\left(\frac{|\mathring{x}_i|\sqrt{T}}{\epsilon}\right)} \,.$$

(The proof of Theorem 7 makes it clear that we can use the same T for all i.) Summing this inequality across all n coordinates then gives the regret bound stated in the theorem. □

The following theorem presents a stronger negative result for Follow-the-Regularized-Leader algorithms with a fixed regularizer: for any such algorithm that guarantees origin-regret at most ϵ_T after T rounds, worst-case regret with respect to any point outside $[-\epsilon_T, \epsilon_T]$ grows linearly with T.

Theorem 9. *Consider a Follow-The-Regularized-Leader algorithm that sets*

$$x_t = \arg\min_x \left(g_{1:t-1} x + \psi_T(x)\right)$$

where ψ_T is a convex, non-negative function with $\psi_T(0) = 0$. Let ϵ_T be the maximum origin-regret incurred by the algorithm on a sequence of T gradients. Then, for any $\mathring{x}$ with $|\mathring{x}| > \epsilon_T$, there exists a sequence of T gradients such that the algorithm's regret with respect to $\mathring{x}$ is at least $\frac{T-1}{2}(|\mathring{x}| - \epsilon_T)$.

In fact, it is clear from the proof that the above result holds for any algorithm that selects x_{t+1} purely as a function of $g_{1:t}$ (in particular, with no dependence on t).

6 Future Work

This work leaves open many interesting questions. It should be possible to apply our techniques to problems that do have constrained feasible sets; for example, it is natural to consider the unconstrained experts problem on the positive orthant. While we believe this extension is straightforward, handling arbitrary non-axis-aligned constraints will be more difficult. Another possibility is to develop an algorithm with bounds in terms of H rather than T that doesn't use a guess and double approach.

References

[1] Jacob Abernethy, Peter L. Bartlett, Alexander Rakhlin, and Ambuj Tewari. Optimal strategies and minimax lower bounds for online convex games. In *COLT*, 2008.

[2] Amit Agarwal, Elad Hazan, Satyen Kale, and Robert E. Schapire. Algorithms for portfolio management based on the Newton method. In *ICML*, 2006.

[3] Nicolò Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, New York, NY, USA, 2006. ISBN 0521841089.

[4] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. In *COLT*, 2010.

[5] Elad Hazan and Satyen Kale. Extracting certainty from uncertainty: Regret bounded by variation in costs. In *COLT*, 2008.

[6] Elad Hazan and Satyen Kale. On stochastic and worst-case models for investing. In *Advances in Neural Information Processing Systems 22*. 2009.

[7] Robert A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1987.

[8] Jyrki Kivinen and Manfred Warmuth. Exponentiated Gradient Versus Gradient Descent for Linear Predictors. *Journal of Information and Computation*, 132, 1997.

[9] Todd K. Leen and Genevieve B. Orr. Optimal stochastic search and adaptive momentum. In *NIPS*, 1993.

[10] H. Brendan McMahan and Matthew Streeter. Adaptive bound optimization for online convex optimization. In *COLT*, 2010.

[11] Barak Pearlmutter. Gradient descent: Second order momentum and saturating error. In *NIPS*, 1991.

[12] Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2012.

[13] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, 2003.