Online Convex Programming and Generalized Infinitesimal Gradient Ascent

Martin Zinkevich

February 2003
CMU-CS-03-110

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Convex programming involves a convex set $F \subseteq \mathbb{R}^n$ and a convex function $c : F \rightarrow \mathbb{R}$. The goal of convex programming is to find a point in F which minimizes c. In this paper, we introduce online convex programming. In online convex programming, the convex set is known in advance, but in each step of some repeated optimization problem, one must select a point in F before seeing the cost function for that step. This can be used to model factory production, farm production, and many other industrial optimization problems where one is unaware of the value of the items produced until they have already been constructed. We introduce an algorithm for this domain, apply it to repeated games, and show that it is really a generalization of infinitesimal gradient ascent, and the results here imply that generalized infinitesimal gradient ascent (GIGA) is universally consistent.

Keywords: multiagent learning, online algorithms

1 Introduction

Imagine a farmer who decides what to plant each year. She has certain restrictions on her resources, both land and labour, as well as restrictions on the output she is allowed to produce. How can she select which crops to grow without knowing in advance what the prices will be? Here we present an algorithm based on gradient descent which will earn her almost as much as she could given that she knew all prices in advance but produced the same amount of each crop year after year. This is an example of an online convex programming problem.

Online convex programming is a generalization of the well-studied experts problem [9, 23]. Imagine that one has n experts, each of which has a plan at each step with some cost. At each round, one selects a probability distribution over experts. If $x \in \mathbb{R}^n$ is defined such that x_i is the probability that one selects expert i, then the set of all probability distributions is a convex set. Also, the cost function on this set is linear, and therefore convex.

In this paper, we present an algorithm for general convex functions based on gradient descent. The algorithm applies gradient descent in $\mathbb{R}^n$, and then moves back to the set of feasible points. There are three advantages to this algorithm. The first is that gradient descent is a simple, natural algorithm that is widely used, and studying its behavior is of intrinsic value. Secondly, this algorithm is more general than the experts setting, in that it can handle an arbitrary sequence of convex functions, which has yet to be solved. Finally, in online linear programs this algorithm can in some circumstances perform better than an experts algorithm. While the bounds on the performance of most experts algorithms depends on the number of experts, these bounds are based on other criterion which may sometimes be lower. This relationship is discussed further in Section 4, and further comments on related work can be found in Section 5. The main theorem is stated and proven in Section 2.1.

The algorithm that motivated this study was infinitesimal gradient ascent [25], which is an algorithm for repeated games. First, this result shows that infinitesimal gradient ascent is universally consistent [11], and secondly it shows that GIGA, a nontrivial extension developed here of infinitesimal gradient ascent to games with more than two actions, is universally consistent. GIGA is defined in Section 3.3 and is proven universally consistent in Section 3.4.

Also, Bansal et al [2] use the results of this paper in the oblivious routing domain.

2 Online Convex Programming

Definition 1 *A set of vectors $S \subseteq \mathbb{R}^n$ is **convex** if for all $x, x' \in S$, and all $\lambda \in [0,1]$, $\lambda x + (1-\lambda)x' \in S$.*

Definition 2 *For a convex set F, a function $f : F \to \mathbb{R}$ is **convex** if for all $x, y \in F$, for all $\lambda \in [0,1]$,*

$$\lambda f(x) + (1-\lambda)f(y) \geq f(\lambda x + (1-\lambda)y)$$

If one were to imagine a convex function $\mathbb{R}^2 \to \mathbb{R}$, where the function described the altitude, then the function would look like a valley.

Definition 3 *A* ***convex programming problem*** *consists of a convex feasible set F and a convex cost function $c : F \to \mathbb{R}$. The* ***optimal solution*** *is the solution that minimizes the cost.*

An example of a convex programming problem is that of a farmer who knows the restrictions on labor and land before she begins, and also knows the demand for her goods in advance.

Suppose that the farmer is not aware of the demand for her products before she begins. She knows that the corresponding cost function is convex, but she is unaware of its actual values. After the year is over and she has sold her items, she then becomes aware of her profit, and can use this to plan the next year. This is an instance of an online convex programming problem.

Definition 4 *An* ***online convex programming problem*** *consists of a feasible set $F \subseteq \mathbb{R}^n$ and an infinite sequence $\{c^1, c^2, \dots\}$ where each $c^t : F \to \mathbb{R}$ is a convex function.*

At each time step t, an ***online convex programming algorithm*** *selects a vector $x^t \in F$. After the vector is selected, it receives the cost function c^t.*

Because all information is not available before decisions are made, online algorithms do not reach "solutions", but instead achieve certain goals. See Section 2.1.

Define $\|x\| = \sqrt{x \cdot x}$ and $d(x, y) = \|x - y\|$. Throughout the remainder of the paper we will make seven assumptions:

1. The feasible set F is **bounded**. There exists $N \in \mathbb{R}$ such that for all $x, y \in F$, $d(x, y) \leq N$.
2. The feasible set F is **closed**. For all sequences $\{x^1, x^2, \dots\}$ where $x^t \in F$ for all t, if there exists a $x \in \mathbb{R}^n$ such that $x = \lim_{t \to \infty} x^t$, then $x \in F$.
3. The feasible set F is **nonempty**. There exists an $x \in F$.
4. For all t, c^t is differentiable[1].
5. There exists an $N \in \mathbb{R}$ such that for all t, for all $x \in F$, $\|\nabla c^t(x)\| \leq N$.
6. For all t, there exists an algorithm, given x, which produces $\nabla c^t(x)$.
7. For all $y \in \mathbb{R}^n$, there exists an algorithm which can produce $\operatorname{argmin}_{x \in F} d(x, y)$. We define the projection $P(y) = \operatorname{argmin}_{x \in F} d(x, y)$.

Given this machinery, we can describe our algorithm.

Algorithm 1 ***Greedy Projection*** *Select an arbitrary $x^1 \in F$ and a sequence of learning rates $\eta_1, \eta_2, \dots \in \mathbb{R}^+$. In time step t, after receiving a cost function, select the next vector x^{t+1} according to:*

$$x^{t+1} = P\left(x^t - \eta_t \nabla c^t(x^t)\right).$$

The basic principle at work in this algorithm is quite clear if we consider the case where the sequence $\{c^1, c^2, \dots\}$ is constant. In this case, our algorithm is operating in an unchanging valley. The boundary of the feasible set is the edge of the valley. By proceeding along the direction opposite the gradient, we walk down into the valley. By projecting back into the convex set, we skirt the edges of the valley.

[1]Although we make the assumption that c^t is differentiable, the algorithm can also work if there exists an algorithm that, given x, can produce a vector g such that for all y, $g \cdot (y - x) \leq c^t(y) - c^t(x)$.

2.1 Analyzing the Performance of the Algorithm

We measure the performance of an algorithm in comparison to the the best algorithm in hindsight that knows all of the cost functions and selects one fixed vector.

Definition 5 *Given an algorithm A, and a convex programming problem $(F, \{c^1, c^2, \dots\})$, if $\{x^1, x^2, \dots\}$ are the vectors selected by A, then the **cost** of A until time T is*

$$C_A(T) = \sum_{t=1}^{T} c^t(x^t).$$

The cost of a static feasible solution $x \in F$ until time T is

$$C_x(T) = \sum_{t=1}^{T} c^t(x).$$

The regret of algorithm A until time T is

$$R_A(T) = C_A(T) - \min_{x \in F} C_x(T).$$

Our goal is to prove that the average regret of Greedy Projection approaches zero. In order to state our results about bounding the regret of this algorithm, we need to specify some parameters. First, let us define:

$$\begin{aligned} \|F\| &= \max_{x,y \in F} d(x,y) \\ \|\nabla c\| &= \sup_{x \in F, t \in \{1,2,\dots\}} \|\nabla c^t(x)\|. \end{aligned}$$

Here is the first result derived in this paper:

Theorem 1 *If $\eta_t = t^{-1/2}$, the regret of the Greedy Projection algorithm is:*

$$R_G(T) \leq \frac{\|F\|^2 \sqrt{T}}{2} + \left(\sqrt{T} - \frac{1}{2}\right) \|\nabla c\|^2$$

Therefore, $\limsup_{T \to \infty} R_G(T)/T \leq 0$.

The first part of the bound is because we might begin on the wrong side of F. The second part is because we always respond after we see the cost function.

Proof: First we show that without loss of generality, for all t there exists a $g^t \in \mathbb{R}^n$ such that for all x, $c^t(x) = g^t \cdot x$.

First, begin with arbitrary $\{c^1, c^2, \dots\}$, run the algorithm and compute $\{x^1, x^2, \dots\}$. Then define $g^t = \nabla c^t(x^t)$. If we were to change c^t such that for all x, $c^t(x) = g^t \cdot x$, the behavior of the algorithm would be the same. Would the regret be the same?

Because c^t is convex, for all x:

$$c^t(x) \geq (\nabla c^t(x^t)) \cdot (x - x^t) + c^t(x^t).$$

Therefore, for all $x^* \in F$: $c^t(x^*) \geq g^t \cdot (x^* - x^t) + c^t(x^t)$. Thus:

$$\begin{aligned} c^t(x^t) - c^t(x^*) &\leq c^t(x^t) - \left(g^t \cdot (x^* - x^t) + c^t(x^t)\right) \\ &\leq g^t \cdot x^t - g^t \cdot x^* \end{aligned}$$

Thus the regret would be at least as much with the modified sequence of functions.

We define for all t, $y^{t+1} = x^t - \eta_t g^t$. Observe that $x^{t+1} = P(y^{t+1})$. We will attempt to bound the regret of not playing action x^* on round t.

$$\begin{aligned} y^{t+1} - x^* &= (x^t - x^*) - \eta_t g^t \\ (y^{t+1} - x^*)^2 &= (x^t - x^*)^2 - 2\eta_t (x^t - x^*) \cdot g^t + \eta_t^2 \| g^t \|^2 \end{aligned}$$

Observe that in the expression $a^2 - 2ab + b^2$, a^2 is a potential, $2ab$ is the immediate cost, and b^2 is the error (within a factor of $2\eta_t$). We will now begin to fully flush out these properties. For all $y \in \mathbb{R}^n$, for all $x \in F$, $(y - x)^2 \geq (P(y) - x)^2$ [13]. Also, $\| g^t \| \leq \| \nabla c \|$. So

$$\begin{aligned} (x^{t+1} - x^*)^2 &\leq (x^t - x^*)^2 - 2\eta_t (x^t - x^*) \cdot g^t + \eta_t^2 \| \nabla c \|^2 \\ (x^t - x^*) \cdot g^t &\leq \frac{1}{2\eta_t}\left((x^t - x^*)^2 - (x^{t+1} - x^*)^2\right) + \frac{\eta_t}{2} \| \nabla c \|^2 \end{aligned}$$

Now, by summing we get:

$$\begin{aligned} R_G(T) &= \sum_{t=1}^{T} (x^t - x^*) \cdot g^t \\ &\leq \sum_{t=1}^{T} \left(\frac{1}{2\eta_t}\left((x^t - x^*)^2 - (x^{t+1} - x^*)^2\right) + \frac{\eta_t}{2} \| \nabla c \|^2 \right) \\ &\leq \frac{1}{2\eta_1}(x^1 - x^*)^2 - \frac{1}{2\eta_T}(x^{T+1} - x^*)^2 + \frac{1}{2}\sum_{t=2}^{T}\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right)(x^t - x^*)^2 + \frac{\| \nabla c \|^2}{2}\sum_{t=1}^{T}\eta_t \\ &\leq \| F \|^2 \left(\frac{1}{2\eta_1} + \frac{1}{2}\sum_{t=2}^{T}\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right)\right) + \frac{\| \nabla c \|^2}{2}\sum_{t=1}^{T}\eta_t \\ &\leq \| F \|^2 \frac{1}{2\eta_T} + \frac{\| \nabla c \|^2}{2}\sum_{t=1}^{T}\eta_t \end{aligned}$$

Now, if we define $\eta_t = \frac{1}{\sqrt{t}}$, then

$$\begin{aligned} \sum_{t=1}^{T} \eta_t &= \sum_{t=1}^{T} \frac{1}{\sqrt{t}} \\ &\leq 1 + \int_{t=1}^{T} \frac{dt}{\sqrt{t}} \\ &\leq 1 + \left[2\sqrt{t}\right]_1^T \\ &\leq 2\sqrt{T} - 1 \end{aligned}$$

Plugging this into the above equation yields the result. ∎

2.2 Regret Against a Dynamic Strategy

Another possibility for the offline algorithm is to allow a small amount of change. For instance, imagine that the path that the offline algorithm follows is of limited size.

Definition 6 *The **path length** of a sequence $x^1, \dots, x^T$ is:*

$$\sum_{t=1}^{T-1} d(x^t, x^{t+1}).$$

Define $\mathbb{A}(T, L)$ to be the set of sequences with T vectors and a path length less than or equal to L.

Definition 7 *Given an algorithm A and a maximum path length L, the **dynamic regret** $R_A(T, L)$ is:*

$$R_A(T, L) = C_A(T) - \min_{A' \in \mathbb{A}(T,L)} C_{A'}(T).$$

Theorem 2 *If η is fixed, the dynamic regret of the Greedy Projection algorithm is:*

$$R_G(T, L) \quad \leq \quad \frac{7\|F\|^2}{4\eta} + \frac{L\|F\|}{\eta} + \frac{T\eta\|\nabla c\|^2}{2}$$

The proof is in the appendix.

2.3 Lazy Projection

In this section, we define a different algorithm that performs suprisingly well.

Algorithm 2 ***(Lazy Projection)*** *Select an arbitrary $x^1 \in F$ and a sequence of learning rates $\eta_1, \eta_2, \ldots \in \mathbb{R}^+$. Define $y^1 = x^1$. In time step t, after receiving a cost function, define y^{t+1}:*

$$y^{t+1} = y^t - \eta_t \nabla c^t(x^t)$$

and select the vector:

$$x^{t+1} = P(y^{t+1}).$$

Theorem 3 *Given a constant learning rate η, Lazy Projection's regret is:*

$$R_L(T) \leq \frac{\|F\|^2}{2\eta} + \frac{\eta\|\nabla c\|^2 T}{2}$$

The proof is in the appendix.

3 Generalized Infinitesimal Gradient Ascent

In this section, we establish that repeated games are online linear programming problems, and an application of our algorithm is universally consistent.

3.1 Repeated Games

From the perspective of one player, a repeated game is two sets of actions A and Y, and a utility function $u : A \times Y \to \mathbb{R}$. A pair in $A \times Y$ is called a **joint action**. For the example in this section, we will think of a matching game. $A = \{a_1, a_2, a_3\}$, $Y = \{y_1, y_2, y_3\}$, where $u(a_1, y_1) = u(a_2, y_2) = u(a_3, y_3) = 1$, and everywhere else u is zero.

As a game is being played, at each step the player will be selecting an action at random based on past joint actions, and the environment will be selecting an action at random based on past joint actions. We will formalize this later.

A **history** is a sequence of joint actions. $H^t = (A \times Y)^t$ is the set of all histories of length t. Define $H = \bigcup_{t=0}^{\infty} H^t$ to be the set of all finite histories, and for any history $h \in H$, define $|h|$ to be the length of that history. An example of a history is:

$$h = \{(a_3, y_1), (a_1, y_2), (a_2, y_3), (a_2, y_2), (a_3, y_3), (a_1, y_2)\}$$

In order to access the history, we define h_i to be the ith joint action. Thus, $h_3 = (a_2, y_3)$, $h_{1,1} = a_3$ and $h_{6,1} = a_1$. The **utility** of a history $h \in H$ is:

$$u_{total}(h) = \sum_{i=1}^{|h|} u(h_{i,1}, h_{i,2}).$$

The utility of the above example is $u_{total}(h) = 2$. We can define what the history would look like if we replaced the action of the player with a_2 at each time step.

$$h^{*\to a_2} = \{(a_2, y_1), (a_2, y_2), (a_2, y_3), (a_2, y_2), (a_2, y_3), (a_2, y_2)\}$$

Now, $u_{total}(h^{*\to a_2}) = 3$. Thus we would have done better playing this action all the time. The definition of **regret of not playing action** a for all $h \in H$, for all $a \in A$ is:

$$R^{*\to a}(h) = u_{total}(h^{*\to a}) - u_{total}(h)$$

In this example, the regret of not playing action a_2 is $R^{*\to a_2}(h) = 3 - 2 = 1$. This regret of not playing an action need not be positive. For instance, $R^{*\to a_1}(h) = 1 - 2 = -1$. Now, we define the maximum regret, or just **regret**, to be:

$$R(h) = \max_{a \in A} R^{*\to a}(h).$$

Here $R(h) = 1$. The most important aspect of this definition of regret is that regret is a function of the resulting history, independent of the strategies that generated that history.

Now, we introduce the definition of the behavior and the environment. For any set S, define $\Delta(S)$ to be the set of all probabilities over S. For a distribution D and a boolean predicate P, we use the notation $\Pr_{x \in D}[P(x)]$ to indicate the probability that $P(x)$ is true given that x was selected from D.

A **behavior** $\sigma : H \to \Delta(A)$ is a function from histories of past actions to distributions over the next action of the player. An **environment** $\rho : H \to \Delta(Y)$ is a function from the history of past actions to distributions over the next action of the environment. Define $H^\infty = (A \times Y)^\infty$ to be the set of all infinite histories, and $h|_t$ to be the history truncated to the first t rounds. Define $F_{\sigma,\rho} \in \Delta(H^\infty)$ to be the distribution over histories of infinite length when σ and ρ play with each other.

Definition 8 *A behavior σ is* ***universally consistent***[2] *if for any $\epsilon > 0$ there exists a T such that for all ρ:*

$$\Pr_{h \in F_{\sigma,\rho}}\left[\forall t > T, \frac{R(h|_t)}{t} > \epsilon\right] < \epsilon.$$

In other words, after some time, with high probability the average regret never again exceeds ϵ. Observe that this convergence over time is uniform over all environments.

We will need other distributions on histories later, so we define them now. We define $F^T_{\sigma,\rho} \in \Delta(H^T)$ to be the distribution over histories of length T when σ and ρ play each other. In addition to these operations, for all h where $|h| < T$ we define $F^T_{\sigma,\rho}(h) \in \Delta(H^T)$ as the distribution when σ and ρ begin with a history h and then play for $T - |h|$ rounds.

3.2 Oblivious Deterministic Environments

An environment is an oblivious deterministic environment if it plays the same sequence of actions regardless of the actions of the player. We will use this type of environment to bridge the gap between the results in online linear programming and repeated games.

Formally, an environment $\rho : H \to \Delta(Y)$ is an oblivious deterministic environment if there exists a function $r : \{1, 2, \dots\} \to Y$ where for all $h \in H$:

$$\Pr_{y \in \rho(h)}[y = r(|h| + 1)] = 1.$$

3.3 Formulating a Repeated Game as an Online Linear Program

For simplicity, suppose that we consider the case where $A = \{1, \dots, n\}$. Before each time step in a repeated game, we select a distribution over actions. This can be represented as a vector in a n-standard closed simplex, the set of all points $x \in \mathbb{R}^n$ such that for all i, $x_i \geq 0$, and $\sum_{i=1}^n x_i = 1$. Define this to be F.

Since we have a utility u instead of cost c, we will perform gradient ascent instead of descent. The utility u is a linear function when the environment's action becomes known.

Algorithm 3 *(**Generalized Infinitesimal Gradient Ascent**) Choose a sequence of learning rates $\{\eta_1, \eta_2, \dots\}$. Begin with an arbitrary vector $x^1 \in F$. Then for each round t:*

[2]This is the generally accepted definition of universally consistent, appearing in [12, 11, 15, 22]. A less restrictive definition originally appeared in [10].

1. *Play according to x^t: play action i with probability x_i^t.*
2. *Observe the action $h_{t,2}$ of the other player and calculate:*

$$\begin{aligned} y_i^{t+1} &= x_i^t + \eta_t u(i, h_{t,2}) \\ x^{t+1} &= P(y^{t+1}) \end{aligned}$$

where $P(y) = \operatorname{argmin}_{x \in F} d(x, y)$, as before.

In this online convex programming problem, $\|F\| \leq \sqrt{2}$, and $\|\nabla c\| \leq \sqrt{|A|}|u|$, where:

$$|u| = \max_{(a,y) \in A \times Y} u(a, y) - \min_{(a,y) \in A \times Y} u(a, y).$$

Now, we can apply the result about greedy projection to get the following result:

Theorem 4 *If $\eta_t = t^{-1/2}$, the expected regret of GIGA for all oblivious deterministic environments $\rho : H \to \Delta(Y)$, for all $a \in A$, for all T is:*

$$\mathbf{E}_{h \in F_{\sigma,\rho}}[R^{* \to a}(h|_T)] \leq \sqrt{T} + \left(\sqrt{T} - \frac{1}{2}\right)|A||u|^2$$

3.4 Self-Oblivious Behavior

It is important to observe that the above is more a method of constructing a behavior than an actual behavior. By proper simulation, the above can be reformulated into a behavior.

Before we begin, we fix x^1. Then, whenever we see a history $h \in H^t$, we simulate how we would have constructed $x^2, \ldots, x^t$ based on the actions of the environment. Thus, given x^1, the above can be reformulated into $\sigma : H \to \Delta(A)$ for any game.

Moreover, σ here is **self-oblivious**, in that it only depends on the history of actions of the environment. Define $Y^* = \bigcup_{i=0}^{\infty} Y^i$, and $\Pi_2 : H \to Y^*$ such that for all $h \in H$,

$$\Pi_2(h) = \{h_{1,2}, h_{2,2}, h_{3,2}, \ldots, h_{|h|,2}\}$$

Definition 9 *A behavior σ is **self-oblivious** if there exists a function $f : Y^* \to \Delta(A)$ such that for all $h \in H$, $\sigma(h) = f(\Pi_2(h))$.*

Self-oblivious algorithms tend to be robust against adaptive adversaries, those that change their technique based on past actions of the behavior.

GIGA is self-oblivious, in that the strategy in the current time step can be calculated given x^1 (a constant) and the past actions of the environment. It should be noted that not all algorithms are self-oblivious. For instance, Kalai and Vempala [19] describe an algorithm that is not self-oblivious, because it uses a "random seed" at the beginning that an adaptive adversary could learn over time and then use in some settings.

The following lemma compartmentalizes the technique used at the end of [9].

Lemma 1 *Given a self-oblivious behavior σ, if for every $\epsilon > 0$ there exists a T such that, for all deterministic, oblivious environments $\rho : H \to \Delta(Y)$, for all $a \in A$, for all $t > T$:*

$$\mathbf{E}_{h \in F_{\sigma,\rho}}\left[\frac{R^{* \to a}(h|_t)}{t}\right] < \epsilon$$

Then σ is universally consistent. Therefore, GIGA is universally consistent.

The proof is in the appendix.

3.5 Lazy Projection and Fictitious Play

There have been some techniques presented to smooth fictitious play in [10], and here we present a very simple version which has much of the "spirit" of fictitious play.

Algorithm 4 ***(Z Fictitious Play)****Choose $\eta > 0$. At time step t define:*

$$y_i^t = \sum_{i=1}^{t-1} \eta u(i, h_{i,2})$$

In other words, the total reward one would have received if one had played action i over the entire history. Define:

$$x^t = P(y^t) = \mathrm{argmin}_{x \in \Delta(A)} d(x, y^t)$$

Play according to x^t in round t.

This algorithm is an instance of Lazy Projection.

4 Converting Old Algorithms

In this section, in order to compare our work with that of others, we show how one can naïvely translate algorithms for mixing experts into algorithms for online linear programs, and online linear programming algorithms into algorithms for online convex programs. This section is a discussion and no formal proofs are given.

4.1 Formal Definitions

We begin with defining the expert's problem.

Definition 10 *An* ***experts problem*** *is a set of experts $E = \{e_1, \ldots, e_n\}$ and a sequence of cost vectors $c^1, c^2, \ldots$ where for all i, $c^i \in \mathbb{R}^n$.*

On each round t, an ***expert algorithm (EA)*** *first selects a distribution $D^t \in \Delta(E)$, and then observes a cost vector c^t.*

We assume that the EA can handle both positive and negative values. If not, it can be easily extended by shifting the values into the positive range.

Now, we define an abstract online linear programming problem.

Definition 11 *An* ***online linear programming problem*** *is a closed convex polytope $F \subseteq \mathbb{R}^n$ and a sequence of cost vectors $c^1, c^2, \ldots$ where for all i,$c^i \in \mathbb{R}^n$.*

On each round t, an ***online linear programming algorithm (OLPA)*** *first plays a distribution $D^t \in \Delta(F)$, and then observes a cost vector c^t.*

An OLPA can be constructed from an EA, as described below.

Algorithm 5 *Define $v^1, \dots, v^k$ to be the vertices of the polytope for an online linear program. Choose $E = \{e_1, \dots, e_k\}$ to be the experts, one for each vertex.*

On each round t, receive a distribution D^t from the EA, and select vector v^i if expert e_i is selected. Define $c' \in \mathbb{R}^k$ such that $c'_i = c^t \cdot v^i$. Send EA the cost vector $c' \in \mathbb{R}^k$.

The optimal static vector must be a vertex of the polytope, because a linear program always has a solution at a vertex of the polytope. If the original EA can do almost as well as the best expert, this OLPA can do at least as well as the best static vector.

The second observation is that most EA have bounds that depend on the number of experts. The number of vertices of the convex polytope is totally unrelated to the diameter, so any normal expert's bound is incomparable to our bound on Greedy Projection.

There are some EA that begin with a distribution or uneven weighting over the experts. These EA may perform better in this scenario, because that one might be able to tweak the distribution such that it is spread evenly over the space (in some way) and not the experts, giving more weight to lonely vertices and less weight to clustered vertices.

4.2 Converting an OLPA to an Online Convex Programming Algorithm

There are two reasons that the algorithm described above will not work for an online convex program. The first is that an online convex program can have an arbitrary convex shape as a feasible region, such as a circle, which cannot be described as the convex hull of any finite number of points.

The second reason is that a convex function may not have an minimum on the edge of the feasible set. For instance, if $F = \{x : x \cdot x \leq 1\}$ and $c(x) = x \cdot x$, the minimum is in the center of the feasible set.

Now, this first issue is difficult to handle directly[3], so we will simply assume that the OLPA can handle the feasible region of the online convex programming problem. This can be either because that the OLPA can handle an arbitrary convex region as in [19], or because that the convex region of the convex programming problem is a convex polytope.

We handle the second issue by converting the cost function to a linear one. In Theorem 1, we find that the worst case is when the cost function is linear. This assumption depends on two properties of the algorithm; the algorithm is deterministic, and the only property of the cost function c^t that is observed is $\nabla c^t(x^t)$.

Now, we form an Online Convex Programming algorithm.

Algorithm 6 ***(Exact)*** *On each round t, receive D^t from the OLPA, and play $x^t = \mathbf{E}_{X \in D^t}[X]$. Send the OLPA the cost vector $\nabla c^t(x^t)$.*

The algorithm is discrete and only observes the gradient at the point x^t, thus we can assume that the cost function is linear. If the cost function is linear, then:

$$\mathbf{E}_{X \in D^t}[c^t(X)] = c^t(\mathbf{E}_{X \in D^t}[X]).$$

x^t may be difficult to compute, so instead of explicitly calculating x^t, we can sample.

[3]One can approximate a convex region by a series of increasingly complex convex polytopes, but this solution is very undesirable.

Algorithm 7 ***(Approx)*** *Select the number of samples $s_1, s_2, \ldots$ to be taken each round. On each round t, sample $X_1, \ldots, X_{s_t}$ independently from the distribution D^t. Play:*

$$z^t = \frac{1}{s_t}\sum_{i=1}^{s_t} X_i$$

Send OLPA the cost vector $\nabla c^t(z^t)$.

We will bound the worst-case difference between Approx and Exact. There are several difficulties in doing so. The first is that Approx is a randomized algorithm, so although it only eventually takes the derivative at a single point, it has a probability of taking the derivative at one of many points, and so both the value and the derivative matter at those points. Secondly, the long-term strategy of the adversary, how it forces the algorithm to move in certain directions, might depend on the random result. Therefore, we try to separate the random element from the deterministic element in a new game. At round t:

1. Receive D^t from the OLPA.
2. Sample $X_1, \ldots, X_{s_t}$ independently from distribution D^t.
3. Calculate and reveal to the adversary $z^t = \frac{1}{s_t}\sum_{i=1}^{s_t} X_i$.
4. Calculate and reveal to the adversary $x^t = \mathbf{E}_{X \in D^t}[X]$.
5. The adversary selects $g^t, h^t \in \mathbb{R}^n$ where $\| g^t \|, \| h^t \| \leq \| \nabla c \|$.
6. Send the OLPA g^t.

This game updates the OLPA in the same fashion as the Exact algorithm. We define the super regret S as:

$$S(T) = \sum_{t=1}^{T} h^t \cdot (z^t - x^t) + \sum_{t=1}^{T} g^t \cdot x^t - \min_{x^* \in F} \sum_{t=1}^{T} g^t \cdot x^*$$

We can relate this super regret to the regret of both the Exact and Approx algorithms.

The second half of the super regret is the regret of the Exact algorithm on a sequence of linear functions.

$$\mathbf{E}[S(T)] = \mathbf{E}[\sum_{t=1}^{T} h^t \cdot (z^t - x^t)] + R_{Exact}(T)$$

We can bound $\mathbf{E}[h^t \cdot (z^t - x^t)]$ based on the number of samples taken:

$$\begin{aligned} \mathbf{E}[h^t \cdot (z^t - x^t)] &\leq \mathbf{E}[\| \nabla c \| d(z^t, x^t)] \\ \mathbf{E}[h^t \cdot (z^t - x^t)] &\leq \| \nabla c \| \mathbf{E}[d(z^t, x^t)] \end{aligned}$$

Without loss of generality, we assume that $x^t = 0$, $0 \in F$.

$$\mathbf{E}[d(0, z^t)] = \frac{1}{s_t}\mathbf{E}\left[\sqrt{\sum_{i,j} X_i X_j}\right]$$

For a random variable Y:

$$\mathbf{E}[Y] \leq \sqrt{\mathbf{E}[Y^2]}$$

We use this fact to prove that:

$$\begin{aligned}
\mathbf{E}[d(0,z^t)] &\leq \frac{1}{s_t}\sqrt{\mathbf{E}\left[\sum_{i,j} X_i \cdot X_j\right]} \\
\mathbf{E}[d(0,z^t)] &\leq \frac{1}{s_t}\sqrt{\sum_i \mathbf{E}\left[\| X_i \|^2\right]} \\
\mathbf{E}[d(0,z^t)] &\leq \frac{1}{s_t}\sqrt{s_t \| F \|^2} \\
\mathbf{E}[d(0,z^t)] &\leq \frac{\| F \|}{\sqrt{s_t}}
\end{aligned}$$

$$\mathbf{E}[S(T)] = \| \nabla c \| \| F \| \sum_{t=1}^{T} \frac{1}{\sqrt{s_t}} + R_{Exact}(T)$$

By choosing $s_t = t$, $\sum_{t=1}^{T} \frac{1}{\sqrt{s_t}} \leq 2\sqrt{T} - 1$. Thus, by selecting s_t properly:

$$\mathbf{E}[S(T)] = \| \nabla c \| \| F \| (2\sqrt{T} - 1) + R_{Exact}(T)$$

We will now prove that $S(T) \geq R_{Approx}(T)$. Imagine that in the approx algorithm each round the adversary knew in advance the random selection z^t before it selects the cost function c^t. This only increases the regret. In the new game, the adversary selects $g^t = \nabla c^t(z^t)$. Therefore:

$$\begin{aligned}
c^t(z^t) - c^t(x^*) &\leq g^t \cdot (z^t - x^*) \\
&\leq g^t \cdot (z^t - x^t + x^t - x^*) \\
&\leq g^t \cdot (z^t - x^t) + g^t \cdot (x^t - x^*)
\end{aligned}$$

The adversary selects h^t to be a vector of length $\| \nabla c \|$ in the direction of $z^t - x^t$.

$$\begin{aligned}
h^t \cdot (z^t - x^t) &= \| \nabla c \| d(x^t, z^t) \\
&\geq \| g^t \| d(x^t, z^t) \\
&\geq g^t \cdot (z^t - x^t)
\end{aligned}$$

So, finally:

$$\begin{aligned}
c^t(z^t) - c^t(x^*) &\leq h^t \cdot (z^t - x^t) + g^t \cdot x^t - g^t \cdot x^* \\
\sum_{t=1}^{T} c^t(z^t) - c^t(x^*) &\leq \sum_{t=1}^{T} h^t \cdot (z^t - x^t) + \sum_{t=1}^{T} g^t \cdot x^t - \min_{x^* \in F} \sum_{t=1}^{T} g^t \cdot x^* \\
R_{Approx}(T) &\leq S(T)
\end{aligned}$$

Therefore, for the proper number of samples:

$$E[R_{Approx}(T)] \leq R_{Exact}(T) + \| \nabla c \| \| F \| (2\sqrt{T} - 1)$$

5 Related Work

Kalai and Vempala [19] have developed algorithms to solve online linear programming, which is a specific type of online convex programming. They are attempting to make the algorithm behave in a lazy fashion, changing its vector slowly, whereas here we are attempting to be more dynamic, as is highlighted in sections 2.2 and 3.4.

These algorithms were motivated by the algorithm of [25] which applies gradient ascent to repeated games. We extend their algorithm to games with an arbitrary number of actions, and prove universal consistency. There has been extensive work on regret in repeated games and in the experts domain, such as [3, 8, 7, 9, 10, 12, 14, 15, 16, 23]. What makes this work noteworthy in a very old field is that it proves that a widely-used technique in artificial intelligence, gradient ascent, has a property that is of interest to those in game theory. As stated in Section 4, experts algorithms can be used to solve online online linear programs and online convex programming problems, but the bounds may become significantly worse.

There are several studies of online gradient descent and related update functions, for example [5, 20, 17, 21]. These studies focus on prediction problems where the loss functions are convex Bregman divergences. In this paper, we are considering arbitrary convex functions, in problems that may or may not involve prediction.

Finally, in the offline case, [6] have done work on proving that gradient descent and projection for arbitrary Bregman distances converges to the optimal result.

6 Conclusions and Future Work

In this paper, we have defined an online convex programming problem. We have established that gradient descent is a very effective technique on this problem. This work was motivated by trying to better understand the infinitesimal gradient ascent algorithm, and the techniques developed we applied to that problem to establish an extension to infinitesimal gradient ascent that is universally consistent.

The simplicity of the algorithm allows for the expansion of these results into other areas. For instance, here we deal with a Euclidean geometry: what if one considered gradient descent on a noneuclidean geometry, like [1, 24]? Also, the simplicity of GIGA allows for this algorithm to be extended for even stronger results, like WoLF[4].

Acknowledgements

As always, any errors or omissions in the work are the sole responsibility of the author. We would like to thank Pat Riley for great help in developing the algorithm for the case of repeated games, Adam Kalai for improving the proof and bounds of the main theorem, and Michael Bowling, Avrim Blum, Nikhil Bansal, and Manfred Warmuth for their help and suggestions with this research.

References

[1] S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10:251–276, 1998.

[2] N. Bansal, A. Blum, S. Chawla, and A. Meyerson. Online oblivious routing. Submitted, 2003.

[3] D. Blackwell. An analog of the minimax theorem for vector payoffs. *South Pacific J. of Mathematics*, pages 1–8, 1956.

[4] M. Bowling and M. Veloso. Convergence of gradient dynamics with a variable learning rate. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 27–34, 2001.

[5] N. Cesa-Bianchi, P. Long, and M. K. Warmuth. Worst-case quadratic bounds for on-line prediction of linear functions by gradient descent. *IEEE Transactions on Neural Networks*, 7:604–619, 1994.

[6] S. Della Pietra, V. Della Pietra, and J. Lafferty. Duality and auxilary functions for Bregman distances. Technical Report CMU-CS-01-109, Carnegie Mellon University, 1999.

[7] D. Foster. A proof of calibration via Blackwell's approachability theorem. In *Games and Economic Behavior*, volume 29, pages 73–79, 1999.

[8] D. Foster and R. Vohra. Regret in the on-line decision problem. *Games and Economic Behavior*, 29(1):7–35, 1999.

[9] Y. Freund and R. Schapire. Adaptive game playing using multiplicative weights. In *Games and Economic Behavior*, volume 29, pages 79–103, 1999.

[10] D. Fudenberg and D. Levine. Universal consistency and cautious fictitious play. *Journal of Economic Dynamics and Control*, 19:1065–1089, 1995.

[11] D. Fudenberg and D. Levine. *The Theory of Learning in Games*. MIT Press, 1998.

[12] D. Fudenberg and D. Levine. Conditional universal consistency. *Games and Economic Behavior*, 29, 1999.

[13] C. Gentile and M. Warmuth. Proving relative loss bounds for online learning algorithms by the Bregman divergence. In *The 13th Annual Conference on Computational Learning Theory*, June 2000. Tutorial.

[14] J. Hannan. Approximation to bayes risk in repeated play. *Annals of Mathematics Studies*, 39:97–139, 1957.

[15] S. Hart and A. Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68:1127–1150, 2000.

[16] S. Hart and A. Mas-Colell. A general class of adaptive strategies. *Journal of Economic Theory*, 98:26–54, 2001.

[17] M. Herbster and M. K. Warmuth. Tracking the best linear predictor. *Journal of Machine Learning Research*, 1:281–309, 2001.

[18] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, March 1963.

[19] A. Kalai and S. Vempala. Geometric algorithms for online optimization. Technical report, MIT, 2002.

[20] J. Kivinen and M. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132:1–64, 1997.

[21] J. Kivinen and M. Warmuth. Relative loss bounds for multidimensional regression problems. *Machine Learning Journal*, 45:301–329, 2001.

[22] D. Levine. Personal communication, 2003.

[23] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. In *Proceedings of the Second Annual Conference on Computational Learning Theory*, 1989.

[24] R. Mahony and R. Williamson. Prior knowledge and preferential structures in gradient descent algorithms. *Journal of Machine Learning Research*, 1:311–355, 2001.

[25] S. Singh, M. Kearns, and Y. Mansour. Nash convergence of gradient dynamics in general-sum games. In *Proceedings of the Sixteenth Conference in Uncertainty in Artificial Intelligence*, pages 541–548, 2000.

A Proof of Dynamic Bounds

Proof: Define z^t to be the dynamic optimal strategy at time t. As before, we can argue that:

$$\begin{aligned}
R_G(T) &\leq \frac{1}{2}\sum_{t=1}^{T}\frac{1}{\eta}\left((x^t - z^t)^2 - (x^{t+1} - z^t)^2\right) + \frac{\|\nabla c\|^2}{2}\sum_{t=1}^{T}\eta \\
R_G(T) &\leq \frac{1}{2\eta}\sum_{t=1}^{T}\left((x^t)^2 - (x^{t+1})^2\right) + \frac{1}{2}\sum_{t=1}^{T}\frac{1}{\eta}2(x^{t+1} - x^t)\cdot(z^t) + \frac{T\eta\|\nabla c\|^2}{2} \\
R_G(T) &\leq \frac{1}{2\eta}\left((x^1)^2 - (x^{T+1})^2\right) + \frac{1}{\eta}x^{T+1}\cdot z^T - \frac{1}{\eta}x^1\cdot z^1 + \sum_{t=2}^{T}\frac{1}{\eta}(z^{t-1} - z^t)\cdot(x^t) + \frac{T\eta\|\nabla c\|^2}{2}
\end{aligned}$$

Without loss of generality, assume $0 \in F$. Thus for any $a, b \in F$, $\|a\|, \|b\| \leq \|F\|$, and $-\frac{\|F\|^2}{4} \leq a \cdot b \leq \|F\|^2$. So:

$$R_G(T) \leq \frac{7\|F\|^2}{4\eta} + \sum_{t=2}^{T} \frac{1}{\eta} d(z^{t-1}, z^t)\|F\| + \frac{T\eta\|\nabla c\|^2}{2}$$

$$R_G(T) \leq \frac{7\|F\|^2}{4\eta} + \frac{L\|F\|}{\eta} + \frac{T\eta\|\nabla c\|^2}{2}$$

∎

B Proof of Lazy Projection

In analyzing Greedy Projection, we had one potential: the distance from the optimal point. In analyzing Lazy Projection, we have two potentials: the distance from y^t to the optimal point (ideal potential), and the distance to the set F (projection potential). The ideal potential is "good", in the sense that we grow distant from the optimal point when we are performing better than the optimal point. However, the projection potential is "bad", in the sense that the farther we go from F, the more difference there is between the vector we wanted to use (y^t) and the vector we did use (x^t). What makes Lazy Projection work is that these two potentials cancel each other.

For all $y \in \mathbb{R}^n$, for all closed, convex sets $F \subseteq \mathbb{R}^n$, define $d(y, F) = \min_{x \in F} d(y, x)$.

The following two proofs are the subcomponents of our general proof. We prove them in Section B.3.

Lemma 2 ***(ideal potential)*** *For any convex set F and any linear cost sequence c^t, defining y^t as in the definition of Lazy Projection with a fixed learning rate η, and any $x^* \in F$:*

$$\sum_{t=1}^{T} c^t(y^t) - c^t(x^*) \leq \frac{\|F\|^2}{2} - \frac{d(y^{T+1}, x^*)^2}{2\eta} + \frac{T\eta\|\nabla c\|^2}{2}$$

Lemma 3 ***(projection potential)*** *For any convex set F and any linear cost sequence c^t, defining y^t and x^t as in the definition of Lazy Projection:*

$$\sum_{t=1}^{T} c^t(x^t) - c^t(y^t) \leq \frac{d(y^{T+1}, F)^2}{2\eta}$$

In Section B.1, we present an example highlighting the issues of Lemma 3. In Section B.2, we prove the critical connection between the n-dimensional case and the one-dimensional case. In Section B.3, we complete the proofs of Lemma 2, Lemma 3, and Theorem 3.

B.1 A Motivating Example

In order to motivate the following technical lemmas, let us consider an example. Choose $F = \{x \in \mathbb{R} : x \leq a\}$. If $y^t \geq a$, $x^t = P(y^t) = a$. Assume that $\eta = 1$ and the cost function at time t is $c^t(x) = g^t \cdot x$. We will attempt to bound the following difference D:

$$D = \sum_{i=1}^{T} c^t(P(y^t)) - c^t(y^t)$$

D is how much less cost we would have accrued had we played the point y^t instead of x^t. Let us assume that for all t, $y^t > a$. So,

$$D = \sum_{i=1}^{T} c^t(a) - c^t(y^t)$$

Since $g^t = y^{t+1} - y^t$, then:

$$D = \sum_{i=1}^{T} (y^{t+1} - y^t) \cdot (y^t - a)$$

Let us define $z^t = d(y^t, F) = \min_{x \in F} d(y^t, x) = y^t - a$. The equation becomes:

$$D = \sum_{i=1}^{T} (z^{t+1} - z^t) z^t$$

We can use the following lemma here:

Lemma 4 *For all $a, b \in \mathbb{R}$:*

$$(a-b)b \leq \frac{a^2 - b^2}{2}$$

Proof: This is an algebraic manipulation of $0 \leq (a-b)^2$. ∎

Thus, we can form a potential:

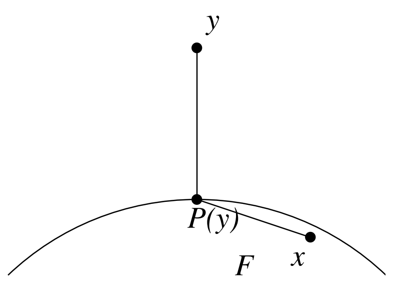
$$\begin{aligned} D &\leq \sum_{i=1}^{T} \frac{(z^{t+1})^2 - (z^t)^2}{2} \\ D &\leq \frac{(z^{T+1})^2 - (z^1)^2}{2} \\ D &\leq \frac{(z^{T+1})^2}{2} \end{aligned}$$

The key step which varies from the general case is that in general $(y^{t+1} - y^t) \cdot (y^t - P(y)) \neq (z^{t+1} - z^t)(z^t)$. However, we prove in the next section that the dot product is always less.

B.2 Geometric Lemmas

This section contains the technical details of the proof.

Figure 1: Lemma 5



Lemma 5 *Given a convex set $F \subseteq \mathbb{R}^n$, if $y \in \mathbb{R}^n$ and $x \in F$, then $(y - P(y)) \cdot (x - P(y)) \leq 0$. In other words, the angle between y,$P(y)$, and x is not acute.*

Proof: We will prove the contrapositive of the theorem. Consider a point $x' \in F$ such that $(y - x') \cdot (x - x') > 0$. We will prove $x' \neq P(y)$. For all $\lambda \in [0,1]$, define:

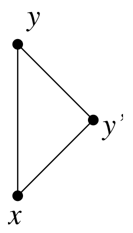
$$z(\lambda) = (1-\lambda)x' + (\lambda)x = x' + \lambda(x - x').$$

We will prove that for small positive values of λ, $z(\lambda)$ is closer to y than x'. Since F is convex, $z(\lambda)$ is in F.

$$\begin{aligned}(y - z(\lambda))^2 &= (y - x' - \lambda(x - x'))^2 \\ (y - z(\lambda))^2 &= \lambda^2(x - x')^2 - 2\lambda(y - x') \cdot (x - x') + (y - x')^2\end{aligned}$$

Observe that for $0 < \lambda < \frac{2(y-x')\cdot(x-x')}{(x-x')^2}$, $(y - z(\lambda))^2 < (y - x')^2$. Thus, $P(y) \neq x'$. ∎

Figure 2: Lemma 6



Lemma 6 *Given* $y, x, y' \in \mathbb{R}^n$, *then*

$$(y-x)\cdot(y'-y) \leq d(y,x)(d(y',x)-d(y,x)).$$

Proof: We begin with a simple example. Suppose that $x, y, y' \in \mathbb{R}^2$, $x = (0,0)$, and $y = (1,0)$. We can prove that the property holds for these three vectors.

$$\begin{aligned} (y-x)\cdot(y'-y) &= (y'_1 - 1) \\ d(y,x)(d(y',x)-d(y,x)) &= \sqrt{(y'_1)^2 + (y'_2)^2} - 1 \\ d(y,x)(d(y',x)-d(y,x)) &\geq (y'_1) - 1 \end{aligned}$$

Now, suppose that for a specific y,x,y', we have proven that the property holds. Then, it will hold for $y+a$, $x+a$, $y'+a$ (a translation), because:

$$\begin{aligned} ((y+a)-(x+a))\cdot((y'+a)-(y+a)) &= (y-x)\cdot(y'-y) \\ &\leq d(y,x)(d(y',x)-d(y,x)) \\ &\leq d(y+a,x+a)(d(y'+a,x+a)-d(y+a,x+a)) \end{aligned}$$

Also, if it holds for y,x,y', then it will hold for ky, kx, and ky' (a scaling), because:

$$\begin{aligned} (ky-kx)\cdot(ky'-ky) &= k^2(y-x)\cdot(y'-y) \\ &\leq k^2 d(y,x)(d(y',x)-d(y,x)) \\ &\leq d(ky,kx)(d(ky',kx)-d(ky,kx)) \end{aligned}$$

Also, the property is invariant under a rotation. Define A^T to be the transpose of the matrix A, and a^T to be the transpose of the vector a. Suppose that R is an orthonormal matrix (where $R^T R = I$). Now, for all $a, b \in \mathbb{R}^n$:

$$(Ra)\cdot(Rb) = (Ra)^T Rb = a^T R^T Rb = a^T b = a \cdot b$$

$$d(Ra, Rb) = \sqrt{(R(a-b))\cdot(R(a-b))} = \sqrt{(a-b)\cdot(a-b)} = d(a,b)$$

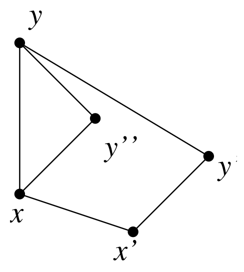
We can now prove that if the property holds for y,x,y', then it will hold for Ry, Rx, Ry'.

$$\begin{aligned} (Ry-Rx)\cdot(Ry'-Ry) &= (R(y-x))\cdot(R(y'-y)) \\ &= (y-x)\cdot(y'-y) \\ &\leq d(y,x)(d(y',x)-d(y,x)) \\ &\leq d(Ry,Rx)(d(Ry',Rx)-d(Ry,Rx)) \end{aligned}$$

Observe that we can think of $\mathbb{R}^2$ as embedded in $\mathbb{R}^n$ without changing distance or dot product. Any three vectors y, x, y' can be obtained from $(0,1,0,\ldots,0)$, $(0,0,0,\ldots,0)$, $(y'_1, y'_2, 0, \ldots, 0)$ using translation, scaling, and rotation. So the property holds for all vectors y, x, y'.

∎

Figure 3: Lemma 7



Lemma 7 *Given $y, x, x', y' \in \mathbb{R}^n$, where $(y - x) \cdot (x' - x) \leq 0$(i.e. the angle is not acute), then:*

$$(y - x) \cdot (y' - y) \leq d(y, x)(d(y', x') - d(y, x))$$

Corollary 1 *Given $y, y' \in \mathbb{R}^n$, if $z = d(y, P(y))$ and $z' = d(y', P(y'))$:*

$$(y - P(y)) \cdot (y' - y) \leq z(z' - z) \leq \frac{(z')^2 - z^2}{2}$$

Proof: If $y = x$, then the result is trivial. Thus, assume $y \neq x$. We begin with an example, as before. We assume $y, x, x', y' \in \mathbb{R}^3$, $y = (1,0,0)$, $x = (0,0,0)$, and $x'_3 = 0$.

Throughout the next part of the proof, we prove that the worst case occurs when $x = x'$. We do this by defining $y'' = y' - x' + x$, and replacing y' with y'' and x' with x. Observe first that $d(y'', x) = d(y', x')$.

Observe that $(y - x) \cdot (y' - y) = y'_1 - 1$, and $(y - x) \cdot (y'' - y) = y'_1 - x'_1 - 1$. Since $(y - x) \cdot (x' - x) = x'_1$, $x'_1 \leq 0$. Thus, $(y - x) \cdot (y' - y) \leq (y - x) \cdot (y'' - y)$, so the relationship only gets tighter as we force $x = x'$. Thus, the property holds for these vectors by Lemma 6.

As in Lemma 6, we can prove that the property is invariant under a transformation, rotation, or scaling.

∎

B.3 Completing the Proof

Now, we complete the proofs. The first part is similar to Theorem 1, and the second is a generalization of the argument in Section B.1.

Proof (of Lemma 2, ideal potential): First, define A (the quantity we are trying to bound):

$$A = \sum_{i=1}^{T} c^t(y^t) - c^t(x^*)$$

For all t there exists a g^t such that $c^t(x) = g^t \cdot x$.

$$A = \sum_{t=1}^{T} g^t \cdot (y^t - x^*)$$

By definition, $y^{t+1} = y^t - \eta g^t$. Similar to Theorem 1:

$$\begin{aligned}
y^{t+1} - x^* &= y^t - x^* - \eta g^t \\
(y^{t+1} - x^*)^2 &= (y^t - x^* - \eta g^t)^2 \\
g^t \cdot (y^t - x^*) &= \frac{(y^t - x^*)^2 - (y^{t+1} - x^*)^2}{2\eta} + \frac{\eta}{2}\| g^t \|^2 \\
g^t \cdot (y^t - x^*) &\leq \frac{(y^t - x^*)^2 - (y^{t+1} - x^*)^2}{2\eta} + \frac{\eta \| \nabla c \|^2}{2}
\end{aligned}$$

Summing, we get:

$$\begin{aligned}
A &\leq \sum_{t=1}^{T} \left(\frac{(y^t - x^*)^2 - (y^{t+1} - x^*)^2}{2\eta} + \frac{\eta \| \nabla c \|^2}{2} \right) \\
A &\leq \frac{(y^1 - x^*)^2 - (y^{T+1} - x^*)^2}{2\eta} + \frac{T\eta \| \nabla c \|^2}{2}
\end{aligned}$$

Since $y^1, x^* \in F$:

$$A \leq \frac{\| F \|^2}{2\eta} - \frac{d(y^{T+1}, F)^2}{2\eta} + \frac{T\eta \| \nabla c \|^2}{2}$$

∎

Proof (of Lemma 3, projection potential): First, define B (the quantity we are trying to bound):

$$\begin{aligned}
B &= \sum_{t=1}^{T} c^t(x^t) - c^t(y^t) \\
B &= \sum_{t=1}^{T} c^t(P(y^t)) - c^t(y^t)
\end{aligned}$$

For all t there exists a g^t such that $c^t(x) = g^t \cdot x$.

$$B = \sum_{t=1}^{T} g^t \cdot (P(y^t) - y^t)$$

Also, $g^t = \frac{y^t - y^{t+1}}{\eta}$. Thus:

$$\begin{aligned} B &= \frac{1}{\eta}\sum_{t=1}^{T}(y^t - y^{t+1}) \cdot (P(y^t) - y^t) \\ B &= \frac{1}{\eta}\sum_{t=1}^{T}(y^t - P(y^t)) \cdot (y^{t+1} - y^t) \end{aligned}$$

Using Corollary 1:

$$\begin{aligned} B &\leq \frac{1}{\eta}\sum_{t=1}^{T}\frac{d(y^{t+1},F)^2 - d(y^t,F)^2}{2} \\ B &\leq \frac{d(y^{T+1},F)^2}{2\eta} \end{aligned}$$

■

Proof (of Theorem 3): Since lazy projection is a deterministic algorithm, and it only considers $\nabla c^t(x^t)$, then the worst case is a linear function. Therefore, we only need to consider linear functions.

$$\begin{aligned} R_G(T) &= \sum_{t=1}^{T} c^t(x^t) - c^t(x^*) \\ R_G(T) &= \sum_{t=1}^{T}(c^t(x^t) - c^t(y^t)) + \sum_{t=1}^{T}(c^t(y^t) - c^t(x^*)) \end{aligned}$$

Thus, by Lemma 2 and Lemma 3:

$$\begin{aligned} R_G(T) &\leq \frac{\|F\|^2}{2\eta} - \frac{d(y^{T+1},F)^2}{2\eta} + \frac{T\eta\|\nabla c\|^2}{2} + \frac{d(y^{T+1},F)^2}{2\eta} \\ &\leq \frac{\|F\|^2}{2\eta} + \frac{T\eta\|\nabla c\|^2}{2} \end{aligned}$$

■

C Proof of Universal Consistency

Our analysis of universal consistency first fixes a behavior σ, a time step T, an $\epsilon > 0$ and an action $a \in A$. Then, we attempt to develop the environment ρ that maximizes the value:

$$\Pr_{h \in F^T_{\sigma,\rho}}[R^{*\to a}(h) > T\epsilon]$$

We will use Doob's Decomposition from the theory of stochastic processes to divide the regret into an "expected" part and a "random" part. We will then bound the expected part of the regret to be lower than the worst case oblivious deterministic environment, due to the fact that the behavior we are studying is self-oblivious. Then, we will bound the random part with an old result from the theory of martingales.

We first introduce some concepts from stochastic processes that will be useful later in the proof.

Definition 12 *A **martingale difference sequence** is a sequence of variables $X_1, X_2, \dots$ such that for all k, for all sequences $X_1, \dots, X_k$:*

$$\mathbf{E}[X_{k+1}|X_1, \dots, X_k] = 0$$

Lemma 8 ***(Azuma's Lemma)*** *[18]: If $X_1, X_2, \dots$ is a martingale difference sequence, and for all i, $|X_i| \leq b$, then:*

$$\Pr\left[\sum_{i=1}^{k} X_i > a\right] \leq exp\left(-\frac{a^2}{2b^2T}\right)$$

Now, Doob's Decomposition allows us to construct a martingale difference sequence out of an arbitrary random sequence $Z_1, Z_2, \dots$ by:

$$Y_i = Z_i - \mathbf{E}[Z_i|Z_{i-1}, \dots, Z_1]$$

Corollary 2 *If $Z_1, Z_2, \dots$ is a random sequence and $|Z_i| \leq b$, then:*

$$\Pr\left[\sum_{i=1}^{k} Z_i - \mathbf{E}[Z_i|Z_{i-1}, \dots, Z_1 > a]\right] \leq exp\left(-\frac{a^2}{4b^2k}\right)$$

In this spirit, we define the functions $V_\sigma : H \to \mathbb{R}$ and $V_\sigma^{rem} : H \to \mathbb{R}$:

$$\begin{aligned} V_\sigma(h) &= \sum_{i=1}^{|h|} \mathbf{E}_{a' \in \sigma(h|_{i-1})}[R^{* \to a}(a', h_{i,2})] \\ V_\sigma^{rem}(h) &= R^{* \to a}(h) - V_\sigma(h) \end{aligned}$$

Lemma 9 *For any self-oblivious σ, for any T, $a \in A$, there exists an oblivious, deterministic environment ρ^* such that, for all $h \in H^T$:*

$$V_\sigma(h) \leq E_{h \in F^T_{\sigma,\rho}}[R^{* \to a}(h)]$$

Proof: Observe that H^T is a finite set. Define $h^* \in H^T$:

$$h^* = \text{argmax}_{h \in H^T} V_\sigma(h)$$

Now, choose some $y' \in Y$, and define ρ^* such that:

$$\begin{aligned}
\Pr_{y \in \rho^*(h)}[y = h^*_{|h|+1,2}] &= 1 \text{ if } |h| < T \\
\Pr_{y \in \rho^*(h)}[y = y'] &= 1 \text{ if } |h| > T
\end{aligned}$$

By construction, ρ^* is a deterministic, oblivious environment. ρ^* will play the actions in $\Pi_2(h^*)$ for the first T rounds. Observe, that **since σ is self-oblivious, the distribution of the actions of σ at time step i is the same when ρ^* plays the actions in $\Pi_2(h^*)$, as when the past history is $h^*|_{i-1}$.** Therefore:

$$\sum_{i=1}^{T} \mathbf{E}_{a' \in \sigma(h^*|_{i-1})}[R^{*\to a}(a', h^*_{i,2})] = \mathbf{E}_{h \in F^T_{\sigma,\rho^*}}[R^{*\to a}(h)].$$

∎

We can use this fact to bound $R^{*\to a}$ with high probability. Define:

$$|u| = \max_{(a,y)\in A\times Y} u(a,y) - \min_{(a,y)\in A\times Y} u(a,y)$$

Lemma 10 *If for a self-oblivious behavior σ and an $a \in A$, for every $\epsilon > 0$ there exists a time t such that for every oblivious, deterministic environment ρ, for every time $T > t$:*

$$\mathbf{E}_{h \in F^T_{\sigma,\rho}}[R^{*\to a}(h)] < T\epsilon$$

Then, for any arbitrary environment ρ':

$$\Pr_{h \in F^T_{\sigma,\rho'}}[R^{*\to a}(h) > 2T\epsilon] < exp\left(\frac{-T\epsilon^2}{8|u|^2}\right)$$

Proof: Choose ρ' to be any arbitrary environment and $T > t$. From Lemma 9, we can define a oblivious, deterministic environment ρ^* such that:

$$V_\sigma(h) \leq E_{h \in F^T_{\sigma,\rho^*}}[R^{*\to a}(h)] < T\epsilon$$

Thus, we have captured the "expected" part of the regret in an arbitrary environment. Define $R^{*\to a} : A \times Y$ such that:

$$R^{*\to a}(a', y) = u(a, y) - u(a', y)$$

Now, for all $i \in \{1, \ldots, T\}$, we define Y_i:

$$Y_i(h) = R^{*\to a}(h_i) - \mathbf{E}_{a' \in \sigma(h|_{i-1})}[R^{*\to a}(a', h_{i,2}))]$$

For all $h \in H$,

$$\begin{aligned} V_\sigma^{rem}(h) &= \sum_{i=1}^{T} Y_i(h) \\ R^{*\to a}(h) &= V_\sigma(h) + \sum_{i=1}^{T} Y_i(h) \\ R^{*\to a}(h) &< T\epsilon + \sum_{i=1}^{T} Y_i(h) \end{aligned}$$

Also, for all $h \in H^{i-1}$:

$$\mathbf{E}_{h' \in F^i_{\sigma,\rho'}(h)}[Y_i(h')] = 0$$

Also, for all i, for all $h \in H$, $|Y_i(h)| \leq 2|u|$. Therefore, by Azuma's Lemma:

$$\begin{aligned} \Pr_{h \in F^T_{\sigma,\rho'}}[\sum_{i=1}^{T} Y_i(h) < T\epsilon] &< exp\left(\frac{-T\epsilon^2}{8|u|^2}\right) \\ \Pr_{h \in F^T_{\sigma,\rho*}}[R^{*\to a}(h) < 2T\epsilon] &< exp\left(\frac{-T\epsilon^2}{8|u|^2}\right) \end{aligned}$$

■

Lemma 11 *If for a behavior σ, for every $\epsilon > 0$ there exists a time t such that for every oblivious, deterministic ρ, for every $a \in A$, for every time $T > t$:*

$$\mathbf{E}_{h \in F^T_{\sigma,\rho}}[R^{*\to a}(h)] < T\epsilon$$

Then, for any arbitrary environment ρ':

$$\Pr_{h \in F^T_{\sigma,\rho'}}[R(h) > 2T\epsilon] < |A| exp\left(\frac{-T\epsilon^2}{8|u|^2}\right)$$

Proof:

$$\begin{aligned} \Pr_{h \in F^T_{\sigma,\rho'}}[R(h) > 2T\epsilon] &= \Pr_{h \in F^T_{\sigma,\rho'}}[\forall a \in A, R^{*\to a}(h) > 2T\epsilon] \\ &\leq \sum_{a \in A} \Pr_{h \in F^T_{\sigma,\rho'}}[R^{*\to a} < 2T\epsilon] \\ &\leq |A| exp\left(\frac{-T\epsilon^2}{8|u|^2}\right) \end{aligned}$$

■

Proof (of Lemma 1): For a given $\epsilon > 0$, we wish to find t''' such that:

$$\Pr_{h \in F_{\sigma,\rho}}[\exists T > t''', R(h|_T) > T\epsilon] < \epsilon$$

We begin by decomposing the infinite sequence of events. Now, for all t:

$$\begin{aligned}\Pr_{h \in F_{\sigma,\rho}}[\exists T > t, R(h|_T) > T\epsilon] &\leq \sum_{T=t+1}^{\infty} \Pr_{h \in F_{\sigma,\rho}}[R(h|_T)] \\ &\leq \sum_{T=t+1}^{\infty} \Pr_{h \in F^T_{\sigma,\rho}}[R(h)]\end{aligned}$$

Define $\epsilon' = \epsilon/2$. There exists a t' such that, for all $T > t'$:

$$\Pr_{h \in F^T_{\sigma,\rho}}[R(h) > 2T\epsilon'] < |A| exp\left(\frac{-T(\epsilon')^2}{8|u|^2}\right)$$

Summing we get, for all $t \geq t'$:

$$\Pr_{h \in F_{\sigma,\rho}}[\exists T > t, R(h|_T) > T\epsilon] \leq \sum_{T=t+1}^{\infty} |A| exp\left(\frac{-T(\epsilon)^2}{32|u|^2}\right)$$

This is a geometric series. For simplicity, define r:

$$r = \exp\left(\frac{-(\epsilon)^2}{32|u|^2}\right)$$

The important fact is that $0 < r < 1$. Calculating the sum:

$$\Pr_{h \in F_{\sigma,\rho}}[\exists T > t, R(h|_T) > T\epsilon] \leq \frac{|A| r^{t+1}}{1 - r^{-1}}$$

Define t'':

$$t'' = \frac{1}{\ln r} \ln\left(\frac{(1 - r^{-1})\epsilon}{|A|}\right) - 1.$$

Thus, for all $t > t''$:

$$\frac{|A| r^{t+1}}{1 - r^{-1}} < \epsilon.$$

Thus, if $t''' = \lceil \max(t', t'') \rceil$:

$$\Pr_{h \in F_{\sigma,\rho}}[\exists T > t''', R(h|_T) > T\epsilon] \leq \epsilon$$

∎