No-regret algorithms for structured prediction problems—DRAFT

Geoffrey J. Gordon

Carnegie Mellon University, Pittsburgh, PA
ggordon@cs.cmu.edu

Abstract. No-regret algorithms are a popular class of learning rules which map a sequence of input vectors $x_1, x_2 \ldots$ to a sequence of predictions $y_1, y_2, \ldots$. Unfortunately, most no-regret algorithms assume that the predictions y_t are chosen from a small, discrete set. We consider instead prediction problems where y_t has internal structure: y_t might be a strategy in a game like poker, or a configuration of a data structure like a rebalancing binary search tree. We derive a family of no-regret learning rules, called Lagrangian Hedging algorithms, to take advantage of this structure. Our algorithms are a direct generalization of known no-regret learning rules like weighted majority and regret matching. In addition to proving regret bounds, we demonstrate one of our algorithms learning to play one-card poker.

1 Introduction

We are given a sequence of input vectors $x_1, x_2, \ldots$, with $x_t \in \mathcal{X}$. After seeing x_t, we predict $y_t \in \mathcal{Y}$. Once we make our prediction, the correct answer is revealed in the form of a convex expected-loss function $\ell_t(y_t)$.[1] Just before seeing the Tth example, our loss is therefore

$$L_T = \sum_{t=1}^{T-1} \ell_t(y_t)$$

If we had predicted using some fixed hypothesis $y = h(x)$ instead, then our loss would have been $\sum_{t=1}^{T-1} \ell(h(x_t))$. We say that our *regret* at time T for not having used h is the difference between these two losses:

$$\rho_T(h) = L_T - \sum_{t=1}^{T-1} \ell_t(h(x_t))$$

Positive regret means that the loss for h is smaller than our actual loss—that is, we would rather have used h. If we fix a comparison class $\mathcal{H}$ of prediction rules,

[1] Many problems use loss functions of the form $\ell_t(y_t) = \ell(y_t, y_t^{\text{true}})$, where ℓ is a fixed function such as squared error and y_t^{true} is a target output. The more general notation allows for problems where there may be more than one correct prediction.

then our overall regret is our regret for not having used the best rule $h \in \mathcal{H}$:

$$\rho_T = \sup_{h \in \mathcal{H}} \rho_T(h)$$

No-regret algorithms are a popular class of learning rules which always have small regret no matter what sequence of examples they see. This no-regret property is a strong guarantee: it holds for all comparison hypotheses $h \in \mathcal{H}$, even though we are choosing which h to compare ourselves to *after* seeing x_t and ℓ_t for all t. And, it holds even if x_t and ℓ_t are statistically dependent from trial to trial; such dependence could result from unmeasured covariates, or from the action of an external agent.

Unfortunately, most no-regret algorithms assume that the predictions y_t are chosen from a small, discrete set. This assumption limits their applicability: in many interesting prediction problems the predictions have some internal structure. For example, in a game of poker (see Section 8 below), the prediction must be a valid poker strategy which specifies how to play during the next hand.

So, we consider prediction problems where $\mathcal{Y}$ is a larger set with internal structure, and derive new learning rules—the Lagrangian hedging algorithms—which take advantage of this structure to provide tighter regret bounds and run faster. The LH algorithms are a direct generalization of known no-regret learning rules like weighted majority and regret matching, and they reduce to these rules when choosing from a small discrete set of predictions.

2 Structured prediction problems

We consider prediction problems where the input set is n-dimensional, the output set is d-dimensional, and the comparison class is a set of linear prediction rules. That is, we assume $\mathcal{X} \subset \mathbb{R}^n$, $\mathcal{Y} \subset \mathbb{R}^d$, and $\mathcal{H} \subset \mathbb{R}^{d \times n}$. We also assume that $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{H}$ are bounded: if they were not, then we could incur unbounded regret on a single trial. For convenience we assume $\mathcal{H}$ is closed.

While $\mathcal{H}$ can be a more or less arbitrary compact subset of $\mathbb{R}^{d \times n}$, we assume that we have an efficient description of some functions related to $\mathcal{H}$ so that we can implement the algorithms described below. For example, $\mathcal{H}$ could be a convex set like $\{H \mid AH = B,\ H \geq 0\}$ for matrices A and B, or a set of discrete points like the corners of a hypercube.

If $\mathcal{H}$ is a convex set, there will never be any need for our algorithm to randomize: for any convex loss function ℓ, we have $\ell(E(h)) \leq E(\ell(h))$ by Jensen's inequality, so we can replace any distribution over $\mathcal{H}$ by its expectation without hurting our performance. On the other hand, if $\mathcal{H}$ is not convex our algorithm may need to randomize to achieve low regret: for example, it is impossible for a deterministic algorithm to guarantee less than $\Theta(t)$ regret in t trials if $\mathcal{H} = \{0, 1\}$.

To build a randomized algorithm we will work with the convex hull of $\mathcal{H}$. A point in $\operatorname{conv} \mathcal{H}$ may be interpreted as a probability distribution over the elements of $\mathcal{H}$: by definition, $h \in \operatorname{conv} \mathcal{H}$ means $h = \sum_i p_i h_i$ for some $h_i \in \mathcal{H}$ and some distribution $p \geq 0$ with $\sum_i p_i = 1$. (In fact, there will usually be

several such representations of a given h; any one of them is acceptable.) Our algorithm will pick a distribution over hypotheses by picking a point in $\operatorname{conv}\mathcal{H}$. For convenience of notation we will take $\mathcal{H}$ to be a convex set in the remainder of this paper; if our desired $\mathcal{H}$ is nonconvex we will work with $\operatorname{conv}\mathcal{H}$ and interpret each hypothesis as a probability distribution over elements of the original $\mathcal{H}$.

3 Regret vectors and safe sets

Lagrangian Hedging algorithms maintain their state in a *regret matrix*. This matrix contains information about our actual losses and the gradients of our loss function. If our loss function ℓ_t is a linear function for each t, $\ell_t(y) = c_t \cdot y$, then we can define the regret matrix S_t by the recursion:

$$S_{t+1} = S_t + ({y_t}^{\mathrm{T}} c_t)E - c_t {x_t}^{\mathrm{T}} \tag{1}$$

with the base case $S_1 = 0$. Here E is an arbitrary matrix which satisfies[2] $H \cdot E = 1$ for all $H \in \mathcal{H}$. If necessary we can append constant elements to each H so that such an E exists. If ℓ_t is nonlinear but convex, we can substitute the derivative $\partial\ell_t(y_t)$ for c_t; our regret bounds will still hold [1, p. 54]. In either case, we will write $\mathcal{C}$ for the set of possible cost vectors c_t.

S_t can tell us our regret versus any hypothesis H:

$$\begin{aligned} H \cdot S_t &= \sum_{i=1}^{t-1} ({y_i}^{\mathrm{T}} c_i) H \cdot E - \sum_{i=1}^{t-1} H \cdot (c_i {x_i}^{\mathrm{T}}) \\ &= L_t - \sum_{i=1}^{t-1} {c_i}^{\mathrm{T}} H x_i = \rho_t(H) \end{aligned}$$

This property justifies the name "regret matrix."

We can define a *safe set*, in which our regret is guaranteed to be nonpositive:

$$\mathcal{S} = \{S \mid (\forall H \in \mathcal{H})\, H \cdot S \le 0\} \tag{2}$$

The goal of the Lagrangian Hedging algorithm will be to keep its regret matrix S_t near the safe set $\mathcal{S}$.

Figure 1 shows an example of the safe set for a very simple hypothesis space in $\mathbb{R}^2$. Note that $\mathcal{S}$ is convex: if $H \cdot S_1 \le 0$ and $H \cdot S_2 \le 0$, then $H \cdot (\lambda_1 S_1 + \lambda_2 S_2) \le 0$ for all $\lambda_1, \lambda_2 \ge 0$ with $\lambda_1 + \lambda_2 = 1$. And, $\mathcal{S}$ is a cone: if $H \cdot S \le 0$, then $H \cdot \lambda S \le 0$ for all $\lambda \ge 0$. Finally, if we define another cone

$$\bar{\mathcal{H}} = \{\lambda H \mid H \in \mathcal{H},\, \lambda \ge 0\} \tag{3}$$

then $\bar{\mathcal{H}}$ is also convex and $\bar{H} \cdot S \le 0$ for all $S \in \mathcal{S}$ and $\bar{H} \in \bar{\mathcal{H}}$. In fact, $\mathcal{S}$ and $\bar{\mathcal{H}}$ are *polar cones*, written $\mathcal{S} = \bar{\mathcal{H}}^{\perp}$ or $\mathcal{S}^{\perp} = \bar{\mathcal{H}}$; see [2] for more detail.

[2] We have written $H \cdot E$ to mean $\sum_{ij} H_{ij} E_{ij}$. Another common notation for this "matrix dot product" is $\operatorname{trace}(HE^{\mathrm{T}})$.

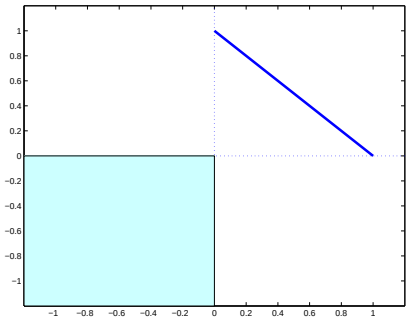

Fig. 1. The set $\mathcal{H} = \{h_1 + h_2 = 1,\ h \geq 0\}$ (thick dark line) and its safe set $\mathcal{S}$ (light shaded region). Note $h \cdot s \leq 0$ for all $h \in \mathcal{H}$ and $s \in \mathcal{S}$.

$S_1 \leftarrow 0$
for $t \leftarrow 1, 2, \ldots$
  $\bar{H}_t \leftarrow f(S_t)$ (*)
  if $\bar{H}_t \cdot E > 0$ **then**
    $H_t \leftarrow \bar{H}_t / (\bar{H}_t \cdot E)$
  else
    $H_t \leftarrow$ arbitrary element of $\mathcal{H}$
  fi
  Observe x_t and set $y_t \leftarrow H_t x_t$
  Observe c_t and compute S_{t+1} from Equation (1)
end

Fig. 2. The gradient form of the Lagrangian Hedging algorithm.

4 Lagrangian hedging

We will present the general Lagrangian hedging algorithm first, then show how to implement it efficiently for specific problems. The general form of the LH algorithm is also called the *gradient form*, as distinguished from the *optimization form*, which is slightly less general but often easier to implement. The optimization form is presented below in Section 5.

At each time step, the LH algorithm chooses its play based on the current regret matrix S_t (as defined in Equation (1)) and the current input vector x_t. The LH algorithm depends on one free parameter, a closed convex potential function $F(S)$ which is defined everywhere in $\mathbb{R}^{d \times n}$. The potential function should be small when S is in the safe set, and large when S is far from the safe set. In order for the algorithm to be well-defined we require

$$F(S) \leq 0 \qquad \forall S \in \mathcal{S} \tag{4}$$

We will impose additional requirements on F later for our regret bounds. For examples of useful potential functions, see Section 7 below. We will write $f(S)$ for an arbitrary subgradient of F; that is, $f(S) \in \partial F(S)$ for all S. (For an introduction to subgradients and convex analysis, see [2].)

The LH algorithm is shown in Figure 2. On each step, it computes $\bar{H}_t = f(S_t)$, then renormalizes to get H_t. To show that the LH algorithm is well-defined, we need to prove that H_t is always a valid hypothesis; Theorem 1, whose proof is given in [3], does so. (Recall that, as described in Section 2, we can replace a nonconvex $\mathcal{H}$ by $\operatorname{conv} \mathcal{H}$ and interpret the elements of $\operatorname{conv} \mathcal{H}$ as probability distributions over the original $\mathcal{H}$.)

Theorem 1. *The LH algorithm is well-defined: given a closed convex hypothesis set $\mathcal{H}$, define $\mathcal{S}$ as in (2) and fix a convex potential function F. If $F(S) \leq 0$ for*

all $S \in \mathcal{S}$, then the LH algorithm with potential F picks hypotheses $H_t \in \mathcal{H}$ for all t.

We can also define a version of the LH algorithm with an adjustable learning rate: if we use the potential function $F(\eta S)$ instead of $F(S)$, the result is equivalent to updating S_t with a learning rate η. The ability to adjust our learning rate will help us obtain regret bounds for some classes of potential functions.

5 The optimization form

Even though we have assumed that we have a convenient representation of our hypothesis space $\mathcal{H}$, it may not be easy to work directly with the safe set $\mathcal{S}$. In particular, it may be difficult to define, evaluate, and differentiate a potential function F which has the necessary properties for our theorems.

For example, a typical choice for F is "squared Euclidean distance from $\mathcal{S}$." If $\mathcal{S}$ is a simple set such as the negative orthant, then this F is relatively easy to work with; but if $\mathcal{S}$ is the safe set for a complicated hypothesis space $\mathcal{H}$ then it is not obvious how to compute $F(S)$ or its derivative efficiently. And, it may also be difficult to prove that F has the curvature properties required for the performance analysis of Theorem 3.

To avoid these difficulties, we can define an alternate form of the Lagrangian Hedging algorithm. This form, called the *optimization form*, defines F in terms of a simpler function W which we will call the *hedging function*. On each step, it computes F and ∂F by solving an optimization problem involving W.

For the algorithm to be well-defined, W should be convex, $\operatorname{dom} W \cap \bar{\mathcal{H}}$ should be nonempty, $W(\bar{H}) \geq 0$ for all $\bar{H}$, and the sets $\{\bar{H} \mid W(\bar{H}) + S \cdot \bar{H} \leq k\}$ should be compact for all S and k. (The last condition is equivalent to saying that W is closed and increases strictly faster than linearly in all directions.) We will impose additional requirements on W later for our regret bounds. For some examples of possible hedging functions, see Section 7.

Define $\bar{\mathcal{H}}$ as in (3). Given W and $\bar{\mathcal{H}}$, we can define the potential function

$$F(S) = \sup_{\bar{H} \in \bar{\mathcal{H}}} (S \cdot \bar{H} - W(\bar{H})) \tag{5}$$

We can compute $F(S)$ by solving (5), but for the LH algorithm we need ∂F instead. As Theorem 2 below shows, there is always an $\bar{H}$ which achieves the maximum in (5)

$$\bar{H} \in \arg\max_{\bar{H} \in \bar{\mathcal{H}}} (S \cdot \bar{H} - W(\bar{H})) \tag{6}$$

and any such $\bar{H}$ is an element of ∂F; so, we can use Equation (6) with $S = S_t$ to compute $\bar{H}_t$ in line $(*)$ of the LH algorithm (Figure 2).

To gain an intuition for Equations (5–6) it may help to consider an example. Suppose that our hypothesis space is the set of probability distributions in d dimensions, as it would be for playing a matrix game or predicting from expert advice. Then, $\mathcal{H}$ is a $(d-1)$-dimensional simplex and $\bar{\mathcal{H}}$ is the positive orthant

in $\mathbb{R}^d$. If we choose the quadratic hedging function $W(\bar{H}) = \|\bar{H}\|_2^2/2$, then the optimization problem (6) will be:

$$\bar{H} = \arg \min_{\bar{H} \in \mathbb{R}^d_+} \|S - \bar{H}\|_2^2/2$$

That is, we can find $\bar{H}$ by projecting S onto $\mathbb{R}^d_+$; this projection is the vector S with its negative elements replaced by zeros, $\bar{H} = [S]_+$. The resulting potential function is one half the sum of the squares of the positive elements of S:

$$F(S) = \| [S]_+ \|_2^2/2$$

The LH algorithm which corresponds to this choice of $\mathcal{H}$ and W is regret matching, and this potential function is the standard one for analyzing this algorithm.

The following theorem shows that the two forms of the LH algorithm are equivalent. Its proof is given in [3].

Theorem 2. *Let W be convex, $\operatorname{dom} W \cap \bar{\mathcal{H}}$ be nonempty, and $W(\bar{H}) \geq 0$ for all $\bar{H}$. Suppose the sets $\{\bar{H} \mid W(\bar{H}) + S \cdot \bar{H} \leq k\}$ are compact for all S and k. Define F as in (5). Then $F(S) \leq 0$ for all $S \in \mathcal{S}$. And, the optimization form of the LH algorithm using the hedging function W is equivalent to the gradient form of the LH algorithm with potential function F.*

6 Theoretical results

Our main theoretical result is a regret bound for the LH algorithm. This bound depends on the curvature of our potential function F, the sizes of the input, output, and hypothesis sets $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{H}$, and the possible slopes $\mathcal{C}$ of our loss functions. Intuitively, F must be neither too curved nor too flat on the scale of the updates to S_t from Equation (1): if F is too curved then ∂F will change too quickly and our hypothesis H_t will jump around a lot, while if F is too flat then we will not react quickly enough to changes in regret.

We will need upper and lower bounds on F. We will assume

$$F(S + \Delta) \leq F(S) + \Delta \cdot f(S) + C\|\Delta\|^2 \tag{7}$$

for all regret matrices S and increments Δ, and

$$F(S) + A \geq \inf_{S' \in \mathcal{S}} B\|S - S'\|^p \qquad \forall S \notin \mathcal{S} \tag{8}$$

Here $\|\cdot\|$ is an arbitrary norm; A, B, and C are positive constants; and $1 \leq p \leq 2$. Equation (7), together with the convexity of F, implies that F is differentiable and that f is its gradient; the LH algorithm is still applicable if F is not differentiable, but its regret bounds are weaker.

The size of our update to S_t (in Equation (1)) depends on the input set $\mathcal{X}$, the output set $\mathcal{Y}$, the cost vector set $\mathcal{C}$, and the matrix E. Rather than bounding each of these items separately, we will assume that there is a constant D so that

$$E(\|S_{t+1} - S_t\|^2 \mid S_t) \leq D \tag{9}$$

Here the expectation is taken with respect to our choice of hypothesis, so the inequality must hold for all possible values of x_t, $y_t = H_t x_t$, and c_t. If we have separate bounds on $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{C}$, and E we can easily combine them to find D.

Finally, we will need a bound on the size of $\mathcal{H}$. We will assume that

$$\|H\|_\circ \leq M \tag{10}$$

for all H in $\mathcal{H}$. Here, $\|\cdot\|_\circ$ is the dual of the norm used in Equation (7). (See [2] for more information about dual norms.) Our theorem then bounds our regret in terms of the above constants; see Appendix A for a proof. Since the bounds are sublinear in t, they show that Lagrangian Hedging is a no-regret algorithm when we choose an appropriate potential F.

Theorem 3. *Suppose that the potential function F is convex and satisfies Equations (4), (7) and (8). Suppose that the problem definition is bounded according to (9) and (10). Then the LH algorithm (Figure 2) achieves expected regret*

$$E(\rho_{t+1}(H)) \leq M((tCD + A)/B)^{1/p} = O(t^{1/p})$$

versus any hypothesis $H \in \mathcal{H}$.

If $p = 1$ the above bound is $O(t)$. But, suppose that we know ahead of time the number of trials t we will see. Define $G(S) = F(\eta S)$, where

$$\eta = \sqrt{A/(tCD)}$$

Then the LH algorithm with potential G achieves regret

$$E(\rho_{t+1}(H)) \leq (2M/B)\sqrt{tACD} = O(\sqrt{t})$$

for any hypothesis $H \in \mathcal{H}$.

In addition to our main result, we have demonstrated how to transfer bounds on the hedging function W to the potential function F. An upper bound on W leads to a lower bound on F, while a lower bound on W yields an upper bound on F; see [3] for more details. These results mean that, in order to analyze or implement the optimization form of the LH algorithm, we never have to build the potential function F or its derivative explicitly.

7 Examples of LH algorithms

7.1 Matrix games and expert advice

The classical applications of no-regret algorithms are learning from expert advice and learning to play a repeated matrix game. These tasks have no difficult structure in $\mathcal{H}$, but we mention them to point out that they are special cases of our general prediction problem. Standard no-regret algorithms such as Freund and Schapire's Hedge [4], Hart and Mas-Colell's regret matching [5], and Littlestone and Warmuth's weighted majority [6] are all special cases of the LH algorithm [3].

7.2 Extensive-form games

Extensive-form games such as poker or bridge are represented by game trees with chance moves and incomplete information. A behavior strategy for a player in an extensive-form game is a function which maps an information state (or equivalently a history of actions and observations) to a distribution over available actions. The number of distinct behavior strategies can be exponential in the size of the game tree; but, by using the sequence form representation of a game [7] we can design algorithms which learn behavior strategies, run in polynomial time, and achieve $O(\sqrt{t})$ regret over t trials. The algorithms described below are the first with all of these properties.

The regret bounds for our algorithms imply that, in the long run, our learner will achieve average cost no worse than its safety value, no matter what strategies our opponents play and without advance knowledge of the payoffs. (Depending on the motivations of our opponents, we may of course do much better.) The proof of this property is identical to the one given for matrix games in [4]; our learning algorithms are the first efficient algorithms to achieve this property in extensive-form games.

We want to learn how to act in an extensive form game through repeated play. To phrase this task as a structured prediction problem, we can set $\mathcal{X} = \{1\}$, $\mathcal{Y}$ to be the set of valid *sequence weight* vectors for our player, and $\mathcal{H} = \mathcal{Y}$. There is one sequence weight y^{sa} for each state-action pair. All weights are positive, and the probability of taking action a in state s is proportional to y^{sa}. The sequence weights satisfy linear constraints which ensure that the payoff for a strategy y is a linear function of y when we hold the strategies of the other players fixed. These constraints are described in more detail in [7]; here we will simply write them as $y \in \mathcal{Y}_{\text{seq}}$.

With the above definitions of $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{H}$, we will repeatedly choose a behavior strategy y_t, find out our cost $y_t \cdot c_t$ and its gradient c_t, and repeat. Note that we are assuming here that we find out the entire cost vector c_t; dealing with reduced feedback is possible, but is beyond the scope of this paper. (For more information, see for example [8, 9].)

Given this setup, we can now develop no-regret algorithms by choosing different hedging functions W. Good choices include the quadratic and entropy hedging functions described above; these result in extensive-form versions of the regret matching and Hedge algorithms. For example, the EF regret matching algorithm runs as follows: given the regret vector

$$s_t = e \sum_{i=1}^{t-1} {y_t}^{\mathrm{T}} c_t - \sum_{i=1}^{t-1} c_t \tag{11}$$

solve the optimization problem

$$\bar{h} = \arg\max_{\bar{h} \in \bar{\mathcal{H}}} \left(s_t \cdot \bar{h} - \|h\|_2^2 / 2 \right) \tag{12}$$

and then normalize $\bar{h}$ to get a feasible sequence weight vector $y_t = h_t \in \mathcal{H}$. The set $\bar{\mathcal{H}}$ is the cone completion of $\mathcal{Y}_{\text{seq}}$; that is, it contains λy for any $\lambda \geq 0$ and

$y \in \mathcal{Y}_{\text{seq}}$. Since $\bar{H}$ can be described by linear equalities and inequalities, the optimization problem (12) is a convex quadratic program and therefore can be solved in polynomial time. The vector e in (11) can be any vector with $e \cdot h = 1$ for all $h \in \mathcal{H}$; for example, we can choose a vector which is zero everywhere except for 1s in components s, a for some initial state s and all actions a.

By evaluating the constants in Theorem 3 we can show regret bounds for the extensive-form algorithms. The proofs are in the long version of this paper [3]. The bound for extensive-form regret matching is:

$$E(\rho_{t+1}(h)) \leq d\sqrt{td}$$

The bound for extensive-form Hedge is $E(\rho_{t+1}(h)) \leq d(2t + \ln d)$ for $\eta = 1$; choosing $\eta = \sqrt{(\ln d)/2t}$ yields regret

$$E(\rho_{t+1}(h)) \leq 2d\sqrt{2t \ln d}$$

So, extensive-form regret matching and extensive-form Hedge are no-regret algorithms.

7.3 Other applications

A large variety of online prediction problems can be cast in our framework. These problems include online convex programming [1, 10, 11], path planning when costs are chosen by an adversary [12], planning in a Markov decision process when costs are chosen by an adversary [13, 14], online pruning of a decision tree [15], and online balancing of a binary search tree [10]. In future work we plan to derive the exact constants in the bounds for various LH algorithms applied to these problems, but in each case the bounds will be polynomial in the dimensionality of the appropriate hypothesis set.

7.4 Comparison to classic no-regret

LH algorithms handle structured prediction problems, that is, problems whose hypothesis class $\mathcal{H}$ is a complex set in some d-dimensional space. Instead of using an LH algorithm, we could consider learning an h by running a standard no-regret algorithm like weighted majority, giving it hypotheses which correspond to the corners $c_1 \ldots c_n$ of $\mathcal{H}$.

Unfortunately, there exist concisely-describable sets $\mathcal{H}$ in d dimensions which have very large numbers of corners. A set like $\{Ah = b,\ h \geq 0\}$ may have exponentially many (in d) extreme points; worse yet, if $\mathcal{H}$ has a curved boundary, it will have infinitely many extreme points. Having a large number of corners in $\mathcal{H}$ hurts us two ways: both the running time and the regret bounds of the standard no-regret algorithms depend on the number of hypotheses. So, if we try to use a standard no-regret algorithm when $\mathcal{H}$ has too many corners we are likely to see poor performance in terms of both runtime and regret.

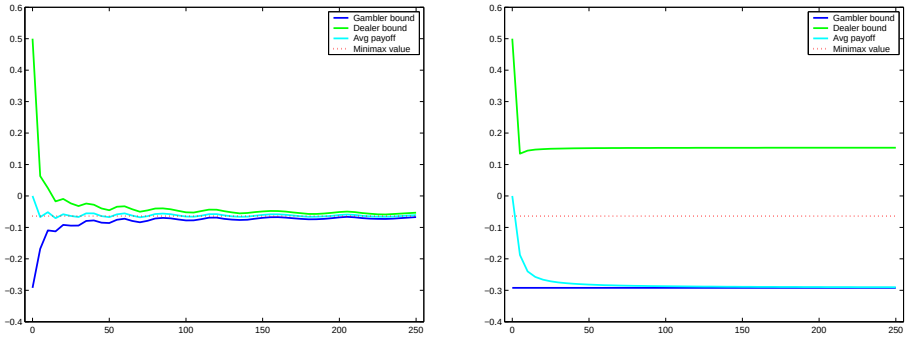

Fig. 3. Performance in self-play (left) and against a fixed opponent (right).

8 Experiments

To demonstrate that our theoretical bounds translate to good practical performance, we implemented the extensive-form regret matching algorithm of Section 7.2 and used it to learn policies for the game of one-card poker. In one-card poker, two players (called the *gambler* and the *dealer*) each ante \$1 and receive one card from a 13-card deck. The gambler bets first, adding either \$0 or \$1 to the pot. Then the dealer gets a chance to bet, again either \$0 or \$1. Finally, if the gambler bet \$0 and the dealer bet \$1, the gambler gets a second chance to bet either \$0 or \$1. If either player bets \$0 when the other has already bet \$1, that player folds and loses his ante. If neither player folds, the higher card wins the pot, resulting in a net gain of either \$1 or \$2 (equal to the other player's ante plus the bet of \$0 or \$1). As mentioned earlier, in contrast to the usual practice in poker we assume that the payoff vector c_t is observable after each hand; the partially-observable extension is beyond the scope of this paper.

One-card poker is a simple game; nonetheless it has many of the elements of more complicated games, including incomplete information, chance events, and multiple stages. And, optimal play requires behaviors like randomization and bluffing.

Figure 3 shows the results of two typical runs: in both panels the dealer is playing our no-regret algorithm. In the left panel the gambler is also playing our no-regret algorithm, while in the right panel the gambler is playing a fixed policy. The horizontal axis shows the number of hands played; the vertical axis shows the average payoff per hand from the dealer to the gambler. The value of the game is indicated with a dotted line. The middle solid curve shows the actual performance of the dealer (who is trying to minimize the payoff).

The upper curve measures the progress of the dealer's learning: after every fifth hand we extracted a strategy y_t^{avg} by taking the average of our algorithm's predictions so far. We then plotted the worst-case value of y_t^{avg}. That is, we plotted the payoff for playing y_t^{avg} against an opponent which knows y_t^{avg} and

is optimized to maximize the dealer's losses. Similarly, the lower curve measures the progress of the gambler's learning.

In the right panel, the dealer quickly learns to win against the non-adaptive gambler. The dealer never plays a minimax strategy (as shown by the fact that the upper curve does not approach the value of the game). Instead, he plays to take advantage of the gambler's weaknesses. In the left panel, the gambler adapts and forces the dealer to play more conservatively; in this case, the limiting strategies for both players are minimax.

9 Discussion and related work

We have presented the Lagrangian hedging algorithms, a family of no-regret algorithms which can handle complex structure in the set of allowable hypotheses or the set of allowable predictions. Examples of problems for which there are efficient LH algorithms include playing extensive form games and planning in an MDP where an adversary controls the cost function. We have proved regret bounds for LH algorithms and demonstrated experimentally that these bounds lead to good predictive performance in practice.

LH algorithms are significant because they have regret bounds with low-order dependences on d, the number of dimensions in $\mathcal{Y}$. This low-order dependence means that the LH algorithms can learn well in high-dimensional prediction problems which would otherwise require an impractical amount of training data. By contrast, the best results which we can prove without taking advantage of the structure of $\mathcal{Y}$ are much worse: as discussed in Section 7.4, both their running time and their regret bounds can be arbitrarily bad.

The prediction problem which LH algorithms solve is a generalization of the so-called *online convex programming* problem [1, 11, 10]. In online convex programming, the input vector is restricted to be $x_t = 1$ (that is, there is no side information). Compared to previous work, our work provides a simple deterministic algorithm; a way to incorporate side information on each trial; the ability to build algorithms from a more general class of potential functions; and a new way of building good potential functions from simpler hedging functions.

Future work includes a no-internal-regret version of the LH algorithm, as well as a bandit-style version. The former will guarantee convergence to a correlated equilibrium in nonzero-sum games, while the latter will allow us to work from incomplete observations of the cost vector (e.g., as might happen in an extensive form game such as poker).

Acknowledgements

Thanks to Amy Greenwald, Martin Zinkevich, and Yoav Shoham for many helpful discussions during the course of this work. This work was supported by AFRL contract F30602–01–C–0219, DARPA's MICA program, and by DARPA's MARS program, NBCH–1020014. The opinions and conclusions are the author's and do not reflect those of the US government or its agencies.

References

1. Gordon, G.J.: Approximate Solutions to Markov Decision Processes. PhD thesis, Carnegie Mellon University (1999)
2. Rockafellar, R.T.: Convex Analysis. Princeton University Press, New Jersey (1970)
3. Gordon, G.J.: No-regret algorithms for structured prediction problems. In preparation (2005)
4. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. In: EuroCOLT 95, Springer-Verlag (1995) 23–37
5. Hart, S., Mas-Colell, A.: A simple adaptive procedure leading to correlated equilibrium. Econometrica **68** (2000) 1127–1150
6. Littlestone, N., Warmuth, M.: The weighted majority algorithm. Technical Report UCSC-CRL-91-28, University of California Santa Cruz (1992)
7. Koller, D., Meggido, N., von Stengel, B.: Efficient computation of equilibria for extensive two-person games. Games and Economic Behaviour **14** (1996)
8. Flaxman, A., Kalai, A., McMahan, H.B.: Online convex optimization in the bandit setting: Gradient descent without a gradient. In: Symposium on Discrete Algorithms (SODA). (2005)
9. Kleinberg, R.: Nearly tight bounds for the continuum-armed bandit problem. In: Advances in Neural Information Processing Systems. Volume 18. (2005)
10. Kalai, A., Vempala, S.: Geometric algorithms for online optimization. Technical Report MIT-LCS-TR-861, MIT (2002)
11. Zinkevich, M.: Online convex programming and generalized infinitesimal gradient ascent. In: 20th ICML. (2003)
12. Takimoto, E., Warmuth, M.: Path kernels and multiplicative updates. In: COLT. (2002)
13. McMahan, H.B., Gordon, G.J., Blum, A.: Planning in the presence of cost functions controlled by an adversary. In: Proc. Int'l Conf. on Machine Learning (ICML). Volume 20. (2003)
14. Even-dar, E., Kakade, S., Mansour, Y.: Experts in a Markov decision process. In: Advances in Neural Information Processing Systems (NIPS). Volume 17. (2004)
15. Helmbold, D.P., Schapire, R.E.: Predicting nearly as well as the best pruning of a decision tree. In: COLT. (1995)

A Proof of main results

This appendix contains the proof of the most important parts of Theorem 3. Due to space constraints the remaining proofs may be found in [3]. Our proof proceeds in three steps: first we will prove a general result about gradient descent (Theorem 4 below) which uses our upper growth bounds, together with the assumption that $S_{t+1} - S_t$ in expectation never points uphill along the gradient of F, to bound the rate of increase of $F(S_t)$. Then we will show that the LH algorithm's choice of hypothesis means that $S_{t+1} - S_t$ satisfies our descent assumption. Finally, we will combine the above results with our lower growth bound on F to show that S_t cannot grow too quickly.

A.1 Bounding the growth of $F(S_t)$

In order to prove our regret bounds we will need our potential function F to have bounded curvature. More precisely, we will require that there exist a function f, a seminorm $\|\cdot\|$, and a constant C so that Equation (7) on p. 6 holds for all S and Δ.[3]

We also need a condition on our updates to S_t: we need them never to point against the gradient of $F(S_t)$. That is, we need

$$E((S_{t+1} - S_t) \cdot f(S_t) \mid S_t) \leq 0 \tag{13}$$

We will call Equation (13) the *generalized Blackwell condition* since it is similar to one of the conditions of Blackwell's approachability theorem. Our first theorem proves a general bound on the growth rate of $F(S_t)$ using conditions (7) and (13).

Theorem 4 (Gradient descent). *Let $F(S)$ and $f(S)$ satisfy Equation (7) using the seminorm $\|\cdot\|$ and the constant C. Let $x_0, x_1, \ldots$ be any sequence of random vectors. Write $S_t = \sum_{i=0}^{t-1} x_i$, and let $E(\|x_t\|^2 \mid S_t) \leq D$ for some constant D. Suppose $E(x_t \cdot f(S_t) \mid S_t) \leq 0$ for all t. Then for all t,*

$$E(F(S_{t+1}) \mid S_1) - F(S_1) \leq tCD$$

Proof. The proof is by induction. For $t = 0$ we have

$$F(S_1) - F(S_1) = 0 \leq 0$$

For $t \geq 1$, assume that

$$E(F(S_t) \mid S_1) \leq F(S_1) + (t-1)CD$$

Then:

$$\begin{aligned} F(S_{t+1}) &= F(S_t + x_t) \\ &\leq F(S_t) + x_t \cdot f(S_t) + C\|x_t\|^2 \\ E(F(S_{t+1}) \mid S_t) &\leq F(S_t) + CD \\ E(F(S_{t+1}) \mid S_1) &\leq E(F(S_t) \mid S_1) + CD \\ E(F(S_{t+1}) \mid S_1) &\leq F(S_1) + (t-1)CD + CD \end{aligned}$$

which is the desired result. The first line above follows from the definition of S_{t+1}; the second, from Equation (7); the third, from taking $E(\cdot \mid S_t)$ on both sides and using the generalized Blackwell condition and our assumption about $\|x_t\|$ to bound the last two terms; the fourth, from taking $E(\cdot \mid S_1)$ on both sides and using the law of iterated expectations; and the last, from the inductive hypothesis.

[3] The text around Equation (7) specifies that F is convex and that $\|\cdot\|$ is a norm, but Theorem 4 holds in the more general case of a nonconvex function and a seminorm. If $\|\cdot\|$ is a norm and F is convex (as will be the case in our application of Theorem 4 below), then Equation (7) implies that F is differentiable everywhere and that f is its gradient.

A.2 The expected change in S_t

We would like to apply Theorem 4 to bound the regret of the Lagrangian Hedging algorithm. To do so, we need to show that the LH algorithm produces a sequence of regret matrices S_t that satisfies the necessary assumptions. We have already assumed (in Equation (9)) that

$$E(\|S_{t+1} - S_t\|^2 \mid S_t) \leq D$$

So, we only need to prove that the sequence S_t satisfies the generalized Blackwell condition, Equation (13). The following lemma does so:

Lemma 1. *The Lagrangian hedging algorithm produces a sequence of regret matrices S_t which satisfies*

$$E((S_{t+1} - S_t) \cdot f_t \mid S_t) \leq 0$$

for all t, where $f_t \in \partial F(S_t)$.

Proof. We will choose f_t to be equal to $\bar{H}_t$ from Figure 2. First note that $kH_t = f_t$ for some $k \geq 0$: in the **then** clause of Figure 2 we have $k = f_t \cdot E$, while in the **else** clause $k = 0$.

Equation (1) tells us that the expected change in the regret matrix is

$$E(S_{t+1} - S_t \mid S_t) = ({c_t}^{\mathrm{T}} H_t x_t)E - c_t {x_t}^{\mathrm{T}}$$

where c_t and x_t are chosen by the opponent but must be independent of H_t. Taking the dot product with H_t yields

$$E((S_{t+1} - S_t) \cdot H_t \mid S_t) = ({c_t}^{\mathrm{T}} H_t x_t)(E \cdot H_t) - {c_t}^{\mathrm{T}} H_t x_t = 0$$

Note that this expected value does not depend on c_t and x_t: the opponent can't influence the expected component of $S_{t+1} - S_t$ along H_t. So, since $kH_t = f_t$ for some $k \geq 0$, we have

$$E((S_{t+1} - S_t) \cdot f_t \mid S_t) = 0$$

which implies the desired conclusion.

A.3 Bounds on the gradient form

In addition to the upper bounds in Equation (7), we will need a lower bound on the growth of $F(S)$ as S gets far away from the safe set $\mathcal{S}$: without such a bound, we would be able to show that $F(S_t)$ doesn't grow too fast, but we would not be able to translate that result to a bound on S_t itself.

Depending on how strong a lower bound we can prove on F, we will get different results about the regret of our algorithm. The strongest results (showing that our average regret decreases as $O(1/\sqrt{t})$) will hold if we can show a quadratic lower bound on F. The bounds will get progressively weaker as our

bounds on F get looser, until the weakest possible lower bound on F (a linear growth rate) gives us the weakest possible upper bound on regret.

To collect all of these results into a single theorem, we will parameterize our lower bound on F by an exponent $1 \leq p \leq 2$, as shown in Equation (8) on p. 6. To make (8) be a non-vacuous lower bound, we will require $\|\cdot\|$ to be a norm rather than a seminorm. (That is, we will require $(\|x\| = 0) \Leftrightarrow (x = 0)$.) With our lower bound we have the following theorem:

Theorem 5. *Suppose that the potential function F is convex and satisfies Equations (4), (7), and (8) for constants A, B, C, and p and a norm $\|\cdot\|$. Suppose that the problem definition is bounded according to (9) and (10). Then the LH algorithm (Figure 2) achieves expected regret*

$$E(\rho_{t+1}(H)) \leq M((tCD + A)/B)^{1/p} = O(t^{1/p})$$

versus any hypothesis $H \in \mathcal{H}$.

Proof. Equations (7) and (9) together with Lemma 1 show that F and the update $S_{t+1} - S_t$ satisfy the assumptions of Theorem 4. So,

$$E(F(S_{t+1}) \mid S_1) - F(S_1) \leq tCD$$

Since $S_1 = 0$ is a fixed constant we can discard the conditioning, and since $S_1 \in \mathcal{S}$ we have $F(S_1) \leq 0$ by Equation (4). Using these facts together with (8) shows that

$$B\, E(\inf_{S \in \mathcal{S}} \|S_{t+1} - S\|^p) \leq tCD + A \tag{14}$$

Write

$$\bar{S} = E(S_{t+1})$$

The function $\inf_{S \in \mathcal{S}} \|\cdot - S\|^p$ is convex (it can be written as $\|\cdot\|^p \,\square\, I_{\mathcal{S}}$), so by Jensen's inequality we have

$$B \inf_{S \in \mathcal{S}} \|\bar{S} - S\|^p \leq tCD + A \tag{15}$$

Now pick any H with $E(\rho_{t+1}(H)) \geq 0$. (If no such H exists, the theorem holds trivially). Our expected regret versus H is

$$E(\rho_{t+1}(H)) = \bar{S} \cdot H = (\bar{S} - S + S) \cdot H$$

for any S. If we choose any $S \in \mathcal{S}$, we know that $S \cdot H \leq 0$. So, for any $S \in \mathcal{S}$,

$$\bar{S} \cdot H \leq (\bar{S} - S) \cdot H \leq \|\bar{S} - S\|\, \|H\|_\circ \leq M\|\bar{S} - S\|$$

by Hölder's inequality and bound (10). Raising to the pth power and choosing the tightest S gives us:

$$(\bar{S} \cdot H)^p \leq M^p \inf_{S \in \mathcal{S}} \|\bar{S} - S\|^p$$

Finally, substituting in Equation (15) gives us

$$(\bar{S} \cdot H)^p \leq M^p(tCD + A)/B$$

which is equivalent to the desired result.