No-regret algorithms for structured prediction problems

Geoffrey J. Gordon

December 21, 2005
CMU-CALD-05-112

School of Computer Science
Carnegie-Mellon University
Pittsburgh, PA 15213

Abstract

No-regret algorithms are a popular class of online learning rules. Unfortunately, most no-regret algorithms assume that the set $\mathcal{Y}$ of allowable hypotheses is small and discrete. We consider instead prediction problems where $\mathcal{Y}$ has internal structure: $\mathcal{Y}$ might be the set of strategies in a game like poker, the set of paths in a graph, or the set of configurations of a data structure like a rebalancing binary search tree; or $\mathcal{Y}$ might be a given convex set (the "online convex programming" problem) or in general an arbitrary bounded set. 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 external-regret matching. In addition to proving regret bounds, we demonstrate one of our algorithms learning to play one-card poker.

Keywords: Lagrangian Hedging algorithms, online learning, multiagent learning, online convex programming, structured prediction problems, extensive-form games, poker, Blackwell approachability.

1 Introduction

In a sequence of trials we are required to pick hypotheses $y_1, y_2, \ldots \in \mathcal{Y}$. After we choose y_t, the correct answer is revealed in the form of a convex expected-loss function $\ell_t(y_t)$.[1] Just before seeing the t^{th} example, our total loss is therefore

$$L_t = \sum_{i=1}^{t-1} \ell_i(y_i)$$

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

$$\rho_t(y) = L_t - \sum_{i=1}^{t-1} \ell_i(y)$$

Positive regret means that the loss for y is smaller than our actual loss—that is, we would rather have used y. Our overall regret is our regret for not having used the best hypothesis $y \in \mathcal{Y}$:

$$\rho_t = \sup_{y \in \mathcal{Y}} \rho_t(y)$$

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 $y \in \mathcal{Y}$, even though we are choosing which y to compare ourselves to *after* seeing ℓ_t for all t. And, it holds even if the loss functions 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, many no-regret algorithms assume that the predictions y_t are probability distributions over a small, discrete set. This assumption limits their applicability: in many interesting prediction problems (such as finding the best pruning of a decision tree, playing poker, balancing an online binary search tree, and planning paths with an adversary) the predictions have some internal structure. For example, in a game of poker (see Section 10 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

[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.

direct generalization of known no-regret learning rules like weighted majority and external-regret matching, and they reduce to these rules when choosing from a small discrete set of predictions.

2 Structured prediction problems

2.1 Problem definition

Our algorithm chooses its prediction at each round from a hypothesis set $\mathcal{Y}$. We assume that $\mathcal{Y}$ is a compact subset of $\mathbb{R}^d$ that has at least two elements.

In classical no-regret algorithms such as weighted majority, $\mathcal{Y}$ is a simplex. The corners of $\mathcal{Y}$ represent pure actions, the interior points of $\mathcal{Y}$ represent probability distributions over pure actions, and the number of corners n is the same as the number of dimensions d. In a structured prediction problem, on the other hand, $\mathcal{Y}$ may have many more extreme points than dimensions, $n \gg d$. For example, $\mathcal{Y}$ could be a convex set like

$$\{y \mid Ay = b,\, y \geq 0\}$$

for some matrix A and vector b (in which case the number of extreme points can be exponential in d), or it could be a sphere (which has infinitely many extreme points), or it could be a set of discrete points like the corners of a hypercube.

The shape of $\mathcal{Y}$ captures the structure in our structured prediction problem. Each point in $\mathcal{Y}$ is a separate hypothesis, but the losses of different hypotheses are related to each other because they are all embedded in the common representation space $\mathbb{R}^d$. This relationship gives us the ability to infer the loss of one hypothesis from the losses of others. For example, consider two Texas Hold'Em strategies which differ only in how aggressively they bet after seeing a particular sequence of play like "Q3 down, no bets, 557 flopped": these strategies will have very similar expected payoffs against any opponent, despite being distinct hypotheses.

It is important to take advantage of available structure in $\mathcal{Y}$. To see why, imagine running a standard no-regret algorithm such as weighted majority on a structured $\mathcal{Y}$: to do so, we must give it hypotheses corresponding to the extreme points $c_1 \ldots c_n$ of $\mathcal{Y}$. Our running time and regret bounds will then depend on the number of extreme points n. If n is exponential in d (as for sets of the form $\{Ay = b,\ y \geq 0\}$), we will have difficulty keeping track of our past loss functions in a reasonable amount of time and space, and our regret bounds may be larger than necessary. If n is infinite (as for spheres), the situation will be even worse: it will be impossible to remember our loss functions at all without some kind of trick, and our regret bounds will be vacuous.

2.2 Reductions which simplify notation

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

To build a randomized algorithm we will allow ourselves to pick hypotheses from the convex hull of $\mathcal{Y}$. We will interpret a point in $\operatorname{conv} \mathcal{Y}$ as a probability distribution over the elements of $\mathcal{Y}$ by decomposing $y = \sum_i p_i y_i$, where $y_i \in \mathcal{Y}$, $p_i \geq 0$, and $\sum_i p_i = 1$. (In fact, there will usually be several such representations of a given y; different ones may yield different regrets, but they will all satisfy our regret bounds below.) For convenience of notation we will take $\mathcal{Y}$ to be a convex set in the remainder of this paper, with the understanding that some elements of $\mathcal{Y}$ may be interpreted as randomized actions.

Our algorithms below are stated in terms of linear loss functions, $\ell_t(y) = c_t \cdot y$. If ℓ_t is nonlinear but convex, we have two options: first, we can substitute the derivative at the current prediction, $\partial \ell_t(y_t)$, for c_t, and our regret bounds will still hold [1, p. 54]. Or, second, we can apply the standard convex programming trick of adding constraints to make our objective linear: for example, if our losses are KL-divergences

$$\ell_t(y) = y \ln \frac{y}{p_t} + (1 - y) \ln \frac{1 - y}{1 - p_t}$$

we can add a new variable z and a new constraint

$$z \geq y \ln y + (1 - y) \ln(1 - y)$$

resulting in a new feasible region $\mathcal{Y}'$.[2] We can then write an equivalent loss function which is linear over $\mathcal{Y}'$:

$$\ell_t(y, z) = z - y \ln p_t - (1 - y) \ln(1 - p_t) \qquad (y, z) \in \mathcal{Y}'$$

In either case we will assume in the remainder of this paper that the loss functions are linear, and we will write $\mathcal{C}$ for the set of possible gradient vectors c_t.

3 Related work

A large number of researchers have studied online prediction in general and online convex programming in particular. From the online prediction literature, the closest related work

[2] Technically, we must also add a vacuous upper bound on z to maintain our assumption of a bounded feasible region.

is that of Cesa-Bianchi and Lugosi [2], which follows in the tradition of an algorithm and proof by Blackwell [3]. Cesa-Bianchi and Lugosi consider choosing predictions from an essentially-arbitrary decision space and receiving outcomes from an essentially-arbitrary outcome space. Together a decision and an outcome determine how a marker $R^t \in \mathbb{R}^d$ will move. Given a potential function G, they present algorithms which keep $G(R_t)$ from growing too quickly. This result is similar in flavor to our Theorem 5, and both Theorem 5 and the results of Cesa-Bianchi and Lugosi are based on Blackwell-like conditions.

The main differences between the Cesa-Bianchi–Lugosi results and ours are the restrictions that they place on their potential functions. They write their potential function as $G(u) = f(\Phi(u))$; they require Φ to be additive (that is, $\Phi(u) = \sum_i \phi_i(u_i)$ for one-dimensional functions ϕ_i), nonnegative, and twice differentiable, and they require $f : \mathbb{R}^+ \mapsto \mathbb{R}^+$ to be increasing, concave, and twice differentiable. These restrictions rule out many of the potential functions used here. The most restrictive requirement is that Φ be additive; for example, unless the set $\bar{\mathcal{Y}}$ can be factored as $\bar{\mathcal{Y}}_1 \times \bar{\mathcal{Y}}_2 \times \ldots \times \bar{\mathcal{Y}}_N$ for one-dimensional sets $\bar{\mathcal{Y}}_1, \bar{\mathcal{Y}}_2, \ldots, \bar{\mathcal{Y}}_N$, potential functions defined via Equation (7) are generally not expressible as $f(\Phi(u))$ for additive Φ. The differentiability requirement rules out potential functions like $[x]_+^2$, which is not twice differentiable at $x = 0$.[3]

Our more general potential functions are what allow us to define no-regret algorithms that work on structured hypothesis spaces like the set of paths through a graph or the set of sequence weights in an extensive-form game. Ours is the first result which allows one to construct such potential functions easily: combining any of a number of well-studied hedging functions (such as negentropy, componentwise negentropy, or squared L_p norms) with an arbitrary compact convex hypothesis set, as described in Section 6, results in a no-regret algorithm. Previous results such as Cesa-Bianchi and Lugosi's provide no guidance in constructing potentials for such hypothesis sets.

In the online convex programming literature, perhaps the best known recent related papers are those of Kalai and Vempala [4] and Zinkevich [5]. The online convex programming problem has a much longer history, though: the first description of the problem and the first algorithm of which we are aware were presented by Hannan in 1957 [6], although Hannan didn't use the name "online convex programming." And, the current author's Generalized Gradient Descent algorithm [1, 7] solves a generalization of the online convex programming problem, although it was not originally presented in those terms: if each of GGD's loss functions $\ell_t(y)$ for $t \geq 1$ is of the form $c_t \cdot y + I(y)$, where I is 0 inside the feasible set and ∞ outside, then GGD solves online convex programs. If in addition GGD's prior loss $\ell_0(y)$ is proportional to $\|y\|_2^2$, then GGD acts like Zinkevich's lazy projection algorithm with a fixed learning rate [8].

[3]Cesa-Bianchi and Lugosi claim (p. 243) that their results apply to $\phi(x) = [x]_+^p$ with $p \geq 2$, but this appears to be a slight error; the Taylor expansion step in the proof on p. 242 requires twice-differentiability and therefore needs $p > 2$. My thanks to Amy Greenwald for pointing this fact out to me.

Compared to the above online convex programming papers, the most important contributions of the current paper are the flexibility of its algorithm and the simplicity and generality of its proof. Ours is the first algorithm based on general potential functions which can solve arbitrary online convex programs.[4] And, our proof contains as special cases most of the common no-regret bounds, including for example those for Hedge and weighted majority: while our overall algorithm is new, by choosing the appropriate potential functions one can reduce it to various well-known algorithms, and our bounds reduce to the corresponding specific bounds.

The flexibility of our algorithm comes from our freedom to choose from a wide range of potential functions; because of this freedom we can design algorithms which force their average regret to zero in a variety of ways. For example, if we define the safe set $\mathcal{S}$ as in Section 4, we can try to decrease two-norm, max-norm, or one-norm distance from $\mathcal{S}$ as rapidly as possible by choosing hedging functions based on $\|y\|_2^2$, negentropy, or componentwise negentropy respectively. The simplicity of the proof results from our use of Blackwell-style approachability arguments; our core result, Theorem 5, takes only half a dozen short equations to prove. This theorem is the first generalization of well-known online learning results such as Cesa-Bianchi and Lugosi's to online convex programming, and it is the most general result of this sort that we know.

More minor contributions include: our bounds are better than those of previous algorithms such as that of Kalai and Vempala, since (unless $p = 1$ in Theorem 3) we do not need to adjust a learning rate based on prior knowledge of the number of trials. And, we are not aware of any prior application of online learning to playing extensive-form games.

In addition to the general papers above, a number of no-regret algorithms for specific online convex programs have appeared in the literature. These include predicting nearly as well as the best pruning of a decision tree [9], reorganizing a binary search tree online so that frequently-accessed items are close to the root [4], and picking paths in a graph with unknown edge costs [10].

4 Regret vectors and safe sets

Lagrangian Hedging algorithms maintain their state in a *regret vector*. This vector contains information about our actual losses and the gradients of our loss functions. Given a loss function $\ell_t(y) = c_t \cdot y$ as described in Section 2, we can define the regret vector s_t by the recursion:

$$s_{t+1} = s_t + (y_t \cdot c_t)u - c_t \tag{1}$$

[4] The current author's GGD and MAP algorithms [1, 7] can both handle a general class of convex potential functions and feasible regions, but they depend either on an adjustable learning rate or on the degree of convexity of the loss functions ℓ_t to achieve sublinear regret.

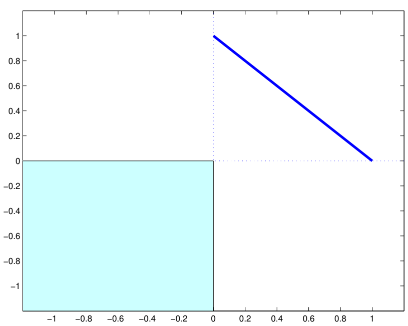

Figure 1: A set $\mathcal{Y} = \{y_1 + y_2 = 1,\, y \geq 0\}$ (thick dark line) and its safe set $\mathcal{S}$ (light shaded region). Note $y \cdot s \leq 0$ for all $y \in \bar{\mathcal{Y}}$ and $s \in \mathcal{S}$, where $\bar{\mathcal{Y}}$ is the positive orthant.

$s_1 \leftarrow 0$
for $t \leftarrow 1, 2, \ldots$
  $\bar{y}_t \leftarrow f(s_t)$ (*)
  if $\bar{y}_t \cdot u > 0$ **then**
    $y_t \leftarrow \bar{y}_t / (\bar{y}_t \cdot u)$
  else
    $y_t \leftarrow$ arbitrary element of $\mathcal{Y}$
  fi
  Observe c_t, compute s_{t+1} from (1)
end

Figure 2: The gradient form of the Lagrangian Hedging algorithm.

with the base case $s_1 = 0$. Here u is an arbitrary vector which satisfies $y \cdot u = 1$ for all $y \in \mathcal{Y}$. If necessary we can append a constant element to each y so that such a u exists.

Given s_t we can compute our regret versus any hypothesis y:

$$y \cdot s_t = \sum_{i=1}^{t-1} (y_i \cdot c_i) y \cdot u - \sum_{i=1}^{t-1} y \cdot c_i = L_t - \sum_{i=1}^{t-1} y \cdot c_i = \rho_t(y)$$

This property justifies the name "regret vector."

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

$$\mathcal{S} = \{s \mid (\forall y \in \mathcal{Y})\, y \cdot s \leq 0\} \tag{2}$$

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

Figure 1 shows an example of the safe set for a very simple hypothesis space in $\mathbb{R}^2$. As is true in general, this example demonstrates that $\mathcal{S}$ is a convex cone: it is closed under positive linear combinations of its elements. If we define another convex cone

$$\bar{\mathcal{Y}} = \{\lambda y \mid y \in \mathcal{Y},\, \lambda \geq 0\} \tag{3}$$

then $\bar{y} \cdot s \leq 0$ for all $s \in \mathcal{S}$ and $\bar{y} \in \bar{\mathcal{Y}}$. In fact, $\bar{\mathcal{Y}}$ is exactly the set of vectors with negative dot products with all of $\mathcal{S}$, and vice versa: the two cones are *polar* to each other, written $\mathcal{S} = \bar{\mathcal{Y}}^\perp$ or $\mathcal{S}^\perp = \bar{\mathcal{Y}}$. See Appendix E for more detail on the properties of polar cones.

5 The Lagrangian Hedging algorithm

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 6. The name "Lagrangian Hedging" comes from the fact that the LH algorithm is a generalization of Freund and Schapire's Hedge algorithm [11], and that its hypothesis can be thought of as a Lagrange multiplier for a constraint which keeps its regret from growing too fast.

At each time step, the LH algorithm chooses its play based on the current regret vector s_t, as defined in Equation (1). The LH algorithm depends on one free parameter, a closed convex potential function $F(s)$ which is defined everywhere in $\mathbb{R}^d$. The potential function should be small when s is in the safe set, and large when s is far from the safe set.

For example, suppose that $\mathcal{Y}$ is the probability simplex in $\mathbb{R}^d$, so that $\mathcal{S}$ is the negative orthant in $\mathbb{R}^d$. (This choice of $\mathcal{Y}$ would be appropriate for playing a matrix game or predicting from expert advice.) For this $\mathcal{Y}$, two possible potential functions are

$$F_1(s) = \ln \sum_i e^{\eta s_i} - \ln d$$

where η is a positive learning rate, and

$$F_2(s) = \sum_i [s_i]_+^2/2$$

where $[s]_+$ is the positive part of s. The potential F_1 will lead to the Hedge [11] and weighted majority [12] algorithms, while the potential F_2 will result in an algorithm called external-regret matching [13, Theorem B]. For a more complicated example of a useful potential function, see Section 9 below.

In order for the LH 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. 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 Appendix E. Such an f is guaranteed to exist since F is finite everywhere.)

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

Theorem 1 *The LH algorithm is well-defined: given a closed convex hypothesis set $\mathcal{Y}$ and a vector u with $u \cdot y = 1$ for all $y \in \mathcal{Y}$, define $\mathcal{S}$ as in (2) and fix a convex potential function F which is everywhere finite. If $F(s) \leq 0$ for all $s \in \mathcal{S}$, then the LH algorithm with potential F picks hypotheses $y_t \in \mathcal{Y}$ 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. Below, the ability to adjust our learning rate will help us obtain regret bounds for some classes of potential functions.

6 The optimization form

Even if we have a convenient representation of our hypothesis space $\mathcal{Y}$, 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 example, a typical choice for F is "squared Euclidean distance from $\mathcal{S}$." If $\mathcal{S}$ is the negative orthant (as it would be for standard experts algorithms), then F is easy to work with: we can separate F into a sum of d simple terms, one for each dimension. On the other hand, if $\mathcal{S}$ is the safe set for a complicated hypothesis space (such as $\mathcal{Y} = \{y \geq 0 \mid Ay + b = 0\}$ for some matrix A and vector b), it is not obvious how to compute $\mathcal{S}$, $F(s)$, or $\partial F(s)$ efficiently: F can have many quadratic pieces with boundaries at many different orientations, and there is generally no way to break F into the sum of a small number of simple terms. For the same reason, 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 work with 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 and the hypothesis set $\mathcal{Y}$. In our example above, where F is squared Euclidean distance from $\mathcal{S}$, the optimization problem is minimum-Euclidean-distance projection: we split s into two orthogonal components, one in $\bar{\mathcal{Y}}$ and one in $\mathcal{S}$. This optimization is easy since we have a compact representation of $\bar{\mathcal{Y}}$. And, knowing the component of s in $\mathcal{S}$ tells us which quadratic piece of F is active, making it easy to compute $F(s)$ and an element of $\partial F(s)$.

For example, two possible hedging functions are

$$W_1(\bar{y}) = \begin{cases} \sum_i \bar{y}_i \ln \bar{y}_i + \ln d & \text{if } \bar{y} \geq 0,\ \sum_i \bar{y}_i = 1 \\ \infty & \text{otherwise} \end{cases} \tag{5}$$

and

$$W_2(\bar{y}) = \sum_i \bar{y}_i^2/2 \tag{6}$$

If $\mathcal{Y}$ is the probability simplex in $\mathbb{R}^d$ (so that $\mathcal{S}$ is the negative orthant in $\mathbb{R}^d$ and we can choose $u = [1, 1, \ldots, 1]^{\mathrm{T}}$), then $W_1(\bar{y}/\eta)$ and $W_2(\bar{y})$ correspond to the potential functions F_1 and F_2 from Section 5 above. So, these hedging functions result in the weighted majority and external-regret matching algorithms respectively. In these examples, since F_1 and F_2 are already simple, W_1 and W_2 are not any simpler. For an example where the hedging function is easy to write analytically but the potential function is much more complicated, see Section 9 below.

For the optimization form of the LH algorithm to be well-defined, W should be convex, $\operatorname{dom} W \cap \bar{\mathcal{Y}}$ should be nonempty, $W(\bar{y}) \geq 0$ for all $\bar{y}$, and the sets $\{\bar{y} \mid W(\bar{y}) + s \cdot \bar{y} \leq k\}$ 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.) Theorem 2 below shows that, under these assumptions, the two forms of the LH algorithm are equivalent. We will impose additional requirements on W later for our regret bounds.

We can now describe the optimization problem which allows us to implement the LH algorithm using W and $\mathcal{Y}$ instead of the corresponding potential function F. Define $\bar{\mathcal{Y}}$ as in (3). Then F is defined to be[5]

$$F(s) = \sup_{\bar{y} \in \bar{\mathcal{Y}}} \left(s \cdot \bar{y} - W(\bar{y})\right) \tag{7}$$

We can compute $F(s)$ by solving (7), but for the LH algorithm we need ∂F instead. As Theorem 2 below shows, there is always a $\bar{y}$ which achieves the maximum in (7):

$$\bar{y} \in \arg\max_{\bar{y} \in \bar{\mathcal{Y}}} \left(s \cdot \bar{y} - W(\bar{y})\right) \tag{8}$$

and any such $\bar{y}$ is an element of ∂F; so, we can use Equation (8) with $s = s_t$ to compute $\bar{y}_t$ in line $(*)$ of the LH algorithm (Figure 2).

To gain an intuition for Equations (7–8), let us look at the example of the external-regret matching algorithm in more detail. Since $\mathcal{Y}$ is the unit simplex in $\mathbb{R}^d$, $\bar{\mathcal{Y}}$ is the positive orthant in $\mathbb{R}^d$. So, with $W_2(\bar{y}) = \|\bar{y}\|_2^2/2$, the optimization problem (8) will be equivalent to

$$\bar{y} = \arg\min_{\bar{y} \in \mathbb{R}^d_+} \frac{1}{2}\|s - \bar{y}\|_2^2$$

That is, $\bar{y}$ is the projection of s onto $\mathbb{R}^d_+$ by minimum Euclidean distance. It is not hard to verify that this projection replaces the negative elements of s with zeros, $\bar{y} = [s]_+$.

[5]This definition is similar to the definition of the convex dual W^* (see Appendix E), but the supremum is over $\bar{y} \in \bar{\mathcal{Y}}$ instead of over all $\bar{y}$. As a result, F and W^* can be very different functions. As pointed out in Appendix B, F can be expressed as the dual of a function related to W: it is $F = (I_{\bar{\mathcal{Y}}} + W)^*$, where $I_{\bar{\mathcal{Y}}}$ is 0 within $\bar{\mathcal{Y}}$ and ∞ outside of $\bar{\mathcal{Y}}$. We state our results in terms of W rather than F^* because W will usually be a simpler function, and so it will generally be easier to verify properties of W.

Substituting this value for $\bar{y}$ back into (7) and using the fact that $s \cdot [s]_+ = [s]_+ \cdot [s]_+$, the resulting potential function is

$$F_2(s) = s \cdot [s]_+ - \sum_i [s_i]_+^2/2 = \sum_i [s_i]_+^2/2$$

as claimed above. This potential function is the standard one for analyzing the external-regret matching algorithm.

Theorem 2 *Let W be convex, $\operatorname{dom} W \cap \bar{\mathcal{Y}}$ be nonempty, and $W(\bar{y}) \geq 0$ for all $\bar{y}$. Suppose the sets $\{\bar{y} \mid W(\bar{y}) + s \cdot \bar{y} \leq k\}$ are compact for all s and k. Define F as in (7). Then F is finite and $F(s) \leq 0$ 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.*

The proof of Theorem 2 is given in Appendix C.

7 Theoretical results

Our main theoretical results are regret bounds for the LH algorithm. The bounds depend on the curvature of our potential function F, the size of the hypothesis set $\mathcal{Y}$, 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 y_t will jump around a lot, while if F is too flat then we will not react quickly enough to changes in regret.

7.1 Gradient form

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

$$F(s + \Delta) \leq F(s) + \Delta \cdot f(s) + C\|\Delta\|^2 \tag{9}$$

for all regret vectors s and increments Δ, and

$$[F(s) + A]_+ \geq \inf_{s' \in \mathcal{S}} B\|s - s'\|^p \tag{10}$$

for all s. Here $\|\cdot\|$ is an arbitrary finite norm, and $A \geq 0$, $B > 0$, $C > 0$, and $1 \leq p \leq 2$ are constants.[6] Equation (9), together with the convexity of F, implies that F is differentiable

[6]The number p has nothing to do with the chosen norm; for example, we could choose $p = 1.5$ but use Euclidean distance (the 2-norm) or even a non-L_p norm.

and that f is its gradient; the LH algorithm is still applicable if F is not differentiable, but its regret bounds are weaker.

We will bound the size of $\mathcal{Y}$ by assuming that

$$\|y\|_\circ \leq M \tag{11}$$

for all y in $\mathcal{Y}$. Here, $\|\cdot\|_\circ$ is the dual of the norm used in Equation (9). (See Appendix E for more information about dual norms.)

The size of our update to s_t (in Equation (1)) depends on the hypothesis set $\mathcal{Y}$, the cost vector set $\mathcal{C}$, and the vector u. We have already bounded $\mathcal{Y}$; rather than bounding $\mathcal{C}$ and u separately, we will assume that there is a constant D so that

$$E(\|s_{t+1} - s_t\|^2 \mid s_t) \leq D \tag{12}$$

Here the expectation is taken with respect to our choice of hypothesis, so the inequality must hold for all possible values of c_t. (The expectation operator is only necessary if we randomize our choice of hypothesis, as would happen if $\mathcal{Y}$ is the convex hull of some non-convex set. If $\mathcal{Y}$ was convex to begin with, we need not randomize, so we can drop the expectation in (12) and below.)

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 the potential function F is convex and satisfies Equations (4), (9) and (10). Suppose that the problem definition is bounded according to (11) and (12). Then the LH algorithm (Figure 2) achieves expected regret*

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

versus any hypothesis $y \in \mathcal{Y}$.

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

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

for any hypothesis $y \in \mathcal{Y}$.

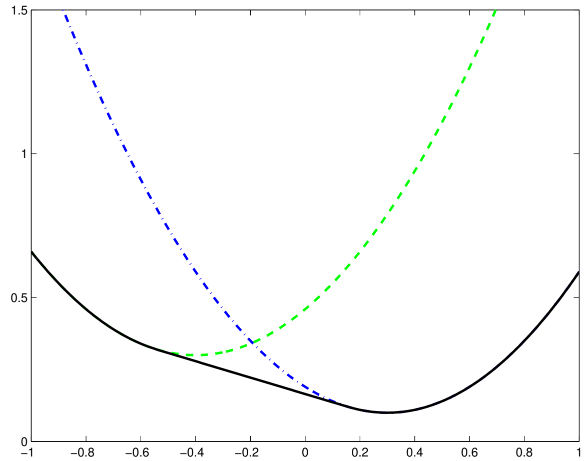

Figure 3: Given two functions F (dashed line) and G (dash-dot), we can define conv $\min(F, G)$ (solid line) to be the pointwise greatest convex function H such that $H(y) \leq \min(F(y), G(y))$ for all y.

7.2 Optimization form

In order to apply Theorem 3 to the optimization form of the LH algorithm, we will show how to transfer bounds on the hedging function W to the potential function F. An upper bound on W will lead to a lower bound on F, while a lower bound on W will yield an upper bound on F. The ability to transfer bounds means that, in order to analyze or implement the optimization form of the LH algorithm, we never have to evaluate the potential function F or its derivative explicitly. Since W and related functions may not be differentiable, we will use the notation of convex analysis to state our bounds; see Appendix E for definitions.

For our upper bound on F, instead of (9) we will assume that for all unnormalized hypotheses $y_0 \in \bar{\mathcal{Y}} \cap \operatorname{dom} \partial W$, for all $s \in \partial W(y_0)$, and for all $y \in \bar{\mathcal{Y}}$,

$$W(y) \geq W(y_0) + (y - y_0) \cdot s + (1/4C)\|y - y_0\|_\circ^2 \tag{13}$$

Here C is the same constant as in Equation (9) and $\|\cdot\|_\circ$ is the dual of the norm from Equation (9). We will also assume that $\bar{\mathcal{Y}} \cap \operatorname{rel\,int} \operatorname{dom} W$ is nonempty; since $\operatorname{rel\,int} \operatorname{dom} W \subseteq \operatorname{dom} \partial W$, this last assumption guarantees that (13) isn't vacuous.

For our lower bound on F, instead of (10) we will assume

$$\operatorname{conv} \min(W(y) - A + I_{\bar{\mathcal{Y}}}(y),\, I_0(y)) \leq B\|y/B\|_\circ^q \qquad \forall y \in \bar{\mathcal{Y}} \tag{14}$$

Here A and B are the same constants as in (10), $\|\cdot\|_\circ$ is the dual of the norm from Equation (10), and $I_K(y)$ is 0 when y is in the set K and ∞ otherwise. The operation conv $\min(F, G)$ is illustrated in Figure 3. The constant q is defined by $\frac{1}{p} + \frac{1}{q} = 1$ where p is the constant from (10). Note that, since $1 \le p \le 2$, we have $2 \le q \le \infty$. As is typical, we will follow the convention

$$|x|^\infty \equiv I_{[-1,1]}(x)$$

So, when $p = 1$, Equation (14) is equivalent to

$$\operatorname{conv} \min(W(y) - A + I_{\bar{\mathcal{Y}}}(y),\, I_0(y)) \le 0 \qquad \forall y \in \bar{\mathcal{Y}} \text{ with } \|y\|_\circ \le B$$

Our main theoretical result about the optimization form of the LH algorithm is that the above bounds on W imply the corresponding bounds on F.

Theorem 4 *Suppose that the hedging function W is closed, convex, nonnegative, and satisfies Equations (13) and (14) with the constants A, B, C, and $2 \le q \le \infty$ and the finite norm $\|\cdot\|_\circ$. Suppose the set $\bar{\mathcal{Y}} \cap \operatorname{rel\,int} \operatorname{dom} W$ is nonempty. Define p so that $\frac{1}{p} + \frac{1}{q} = 1$. Define F as in (7). Then the optimization form of the LH algorithm using hedging function W is equivalent to the gradient form using potential function F, and F satisfies the assumptions of Theorem 3 with constants A, B, C, and p and the norm $\|\cdot\|$.*

Theorem 4 follows directly from Theorems 2 and 9 (proven in Appendices B and C). As an immediate corollary we have that the optimization form satisfies all of the same regret bounds as the gradient form; for example, if the problem definition is bounded by (11) and (12) with constants M and D, Theorem 3 shows that our expected regret is bounded by

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

after t steps versus any hypothesis $y \in \mathcal{Y}$.

One result which is slightly tricky to carry over is the use of learning rates to achieve no regret when $p = 1$. The choice of learning rate and the resulting bound are the same as in Theorem 3, but the implementation is slightly different: to set a learning rate $\eta > 0$, we want to use the potential

$$G(s) = F(\eta s) = \sup_{\bar{y} \in \bar{\mathcal{Y}}} (\eta s \cdot \bar{y} - W(\bar{y}))$$

Using the substitution $\bar{y} \mapsto \bar{y}/\eta$, we have

$$G(s) = \sup_{\bar{y} \in \bar{\mathcal{Y}}} (s \cdot \bar{y} - W(\bar{y}/\eta))$$

since $\bar{y}/\eta \in \bar{\mathcal{Y}}$ whenever $\bar{y} \in \bar{\mathcal{Y}}$. So, to achieve a learning rate η, we just need to replace $W(\bar{y})$ with $W(\bar{y}/\eta)$.

8 Examples

8.1 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 two tasks are essentially equivalent, since they both use the probability simplex

$$\mathcal{Y} = \{y \mid y \geq 0, \textstyle\sum_i y_i = 1\}$$

for their hypothesis set. This choice of $\mathcal{Y}$ has no difficult structure, but we mention it to point out that it is a special case of our general prediction problem. Standard no-regret algorithms such as Freund and Schapire's Hedge [11], Littlestone and Warmuth's weighted majority [12], and Hart and Mas-Colell's external-regret matching [13, Theorem B] are all special cases of the LH algorithm.

For definiteness, we will consider the case of repeated matrix games. On step t we choose a probability distribution y_t over our possible actions. Our opponent plays a mixed strategy z_t over his possible actions, and we receive payoff $\ell_t(y_t) = {z_t}^{\mathrm{T}} M y_t = c_t \cdot y_t$ where M is our payoff matrix. Our problem is to learn how to play well from experience: since we do not know our opponent's payoff matrix, we wish to adjust our own play to achieve high reward against the actual sequence of plays $z_1, z_2, \ldots$ that we observe.

8.1.1 External-regret matching

Perhaps the simplest no-regret algorithm for matrix games is the one we get by taking $W(y) = \|y\|_2^2/2$, which leads to $F(s) = \|\,[s]_+\|_2^2/2$ as described above. The derivative of F is

$$f(s) = [s]_+$$

so at each step we take the positive components of our regret vector, renormalize them to form a probability distribution, and play according to this probability distribution.

Using the Euclidean norm $\|\cdot\|_2$, it is easy to see that our choice of W satisfies Equation (13) with $C = 1/2$ and Equation (14) with $A = 0$, $B = 1/2$, and $p = q = 2$. All elements of the probability simplex $\mathcal{Y}$ are bounded by $\|y\|_2 \leq 1$, so $M = 1$ in Equation (11). And, if our payoff matrix is bounded so that so that $0 \leq M_{ij} \leq 1$, then $c_t \in [0,1]^d$ and $y_t \cdot c_t \in [0,1]$ in (1), so our regret updates are in $[-1,1]^d$. That means that we can take $D = d$ in Equation (12).

Substituting in the above constants, Theorem 3 tells us that the external-regret matching algorithm has regret

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

for any comparison hypothesis $y \in \mathcal{Y}$.

8.1.2 Hedge

Another well-known no-regret algorithm for matrix games is Hedge [11]. To reproduce this algorithm, we can use the potential function

$$F(s) = \ln \sum_i e^{s_i} - \ln d$$

in the gradient form of the LH algorithm. The gradient of F is

$$f_i(s) = e^{s_i} / \textstyle\sum_j e^{s_j}$$

So, at each step we exponentiate the regrets and then renormalize to get a probability distribution. This is exactly the Hedge algorithm: the usual formulation of Hedge says to exponentiate the sum of the loss vectors instead of the regret vector, but since the regret differs from the sum of the losses by a multiple of $u = (1, 1, \ldots, 1)^{\mathrm{T}}$, the difference gets canceled out in the normalizing constant.

For the generalizations of Hedge which we will examine below, it will be helpful to prove our bounds using the optimization form of the LH algorithm. In the optimization form, Hedge uses the entropy hedging function shown in Equation (5). This choice of W is finite only inside $\bar{\mathcal{Y}} = \mathbb{R}^d_+$, so the optimization (7) just computes $W^*(s)$; it is a standard result that the F given above is equal to W^*.

Using the L_1 norm $\|\cdot\|_1$, our choice of W satisfies Equation (13) with $C = 1/2$ and Equation (14) with $A = \ln d$, $B = 1$, $p = 1$, and $q = \infty$. For a proof, see Lemma 10 in Appendix D. All elements of the probability simplex $\mathcal{Y}$ are bounded by $\|y\|_1 \leq 1$, so $M = 1$ in Equation (11). Finally, our regret updates are in $[-1, 1]^d$ and so have max norm no more than 1; so, we can take $D = 1$ in Equation (12).

Substituting in the above constants, Theorems 3 and 4 tell us that the Hedge algorithm with learning rate $\eta = 1$ has

$$E(\rho_{t+1}(y)) \leq t/2 + \ln d$$

for any comparison hypothesis y. If we pick instead $\eta = \sqrt{(2 \ln d)/t}$, the bound becomes

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

This result is similar to well-known bounds on Hedge such as the one obtained by Freund and Schapire [11, section 2.2]. Translated to our notation, Freund and Schapire chose a learning rate of

$$\eta = \ln(1 + \sqrt{(2 \ln d)/t})$$

which is slightly slower than our learning rate. They used this learning rate to prove a regret bound of

$$\sqrt{(2 \ln d)/t} + (\ln d)/t$$

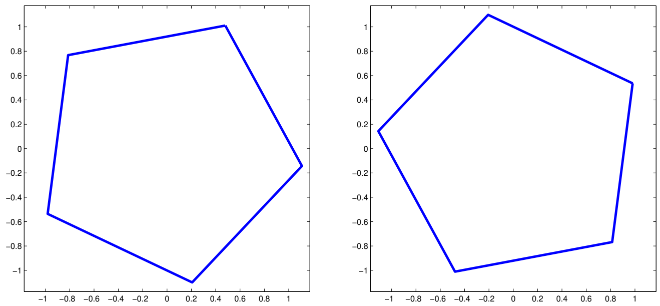

Figure 4: Synthetic example of a structured prediction problem. Left: domain of x. Right: domain of y.

per trial, which is slightly weaker than our bound since it adds a term depending on $1/t$. As $t \to \infty$, the difference in learning rates approaches zero and the $O(1/t)$ term becomes irrelevant, so the two bounds become equivalent.[7]

8.2 A simple synthetic example

This subsection presents a simple synthetic example of a structured prediction problem and an LH algorithm which solves it. Unlike the examples in the previous subsection, there is no obvious way to select a potential function for this problem without either using the techniques described in this paper or moving to a less efficient representation (such as the one where each corner of $\mathcal{Y}$ has a separate regret). In addition, this example demonstrates how to apply the LH algorithm to regression or classification problems: in these problems, each example consists of an input vector x_t together with a target output z_t, and our hypothesis space is a set of functions $\mathcal{Y}$ which map inputs to outputs.

In our synthetic problem, the input examples x_t are drawn from the pentagon $\mathcal{X}$ shown at the left of Figure 4, and the target outputs z_t are either $+1$ or -1. Our predictions are linear functions which map $\mathcal{X}$ into the interval $[-1, 1]$; the set $\mathcal{Y}$ of such functions is the geometric dual of $\mathcal{X}$, which is the pentagon shown on the right of Figure 4. We will use the absolute loss

$$\ell_t(y) = \begin{cases} y \cdot x_t & \text{if } z_t = -1 \\ -y \cdot x_t & \text{if } z_t = +1 \end{cases}$$

or more compactly $\ell_t(y) = -z_t x_t \cdot y$.

Specifying $\mathcal{Y}$ and ℓ_t completely describes our prediction problem. The set $\mathcal{Y}$ is not particularly complicated, but it does not match the hypothesis sets for any of the standard

[7]The extra term in Freund and Schapire's bound appears to be due to the fact that they write the recommended distribution of actions as β^{-S}/Z rather than $\exp(\eta S)/Z$, requiring an extra linearization step $\ln(1 + \beta) \leq \beta$ in their proofs.

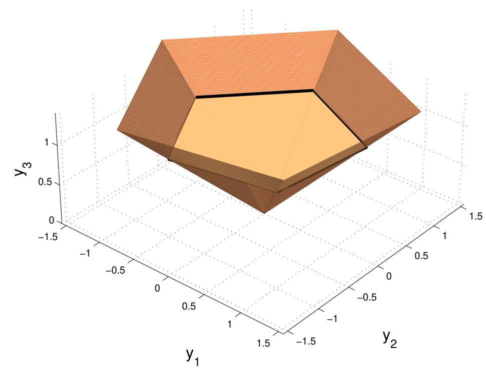

Figure 5: Hypothesis space $\mathcal{Y}$ after including constant component, together with the cone $\bar{\mathcal{Y}}$ containing $\mathcal{Y}$.

no-regret algorithms. So, we will design a Lagrangian Hedging algorithm instead.

In order to construct an LH algorithm we need a vector u with $u \cdot y = 1$ for all $y \in \mathcal{Y}$. Since such a vector doesn't exist for the $\mathcal{Y}$ shown in Figure 4, we will add a dummy dimension to the problem: we will set the third element of y to be 1 for all $y \in \mathcal{Y}$, and add a corresponding third element of 0 onto each x so that the predictions remain unchanged. The modified $\mathcal{Y}$ is shown in Figure 5 as a horizontal pentagon. Figure 5 also shows the boundaries of a cone extending from the origin through $\mathcal{Y}$; this cone is $\bar{\mathcal{Y}}$.

With our modified $\mathcal{Y}$ we can take $u = (0, 0, 1)^{\mathrm{T}}$. So, the only thing left to specify in our LH algorithm is our hedging function W. For simplicity we will pick squared Euclidean norm, $\|\bar{y}\|_2^2/2$. Having chosen a hedging function we can now apply the optimization form of the LH algorithm. The algorithm starts with $s_1 = (0, 0, 0)^{\mathrm{T}}$, then, for each t, executes the following steps:

- Project s_t onto $\bar{\mathcal{Y}}$ by minimum Euclidean distance; call the result $\bar{y}$.
- Normalize $\bar{y}$ to get $y = \bar{y}/(\bar{y} \cdot u)$. (If $\bar{y} \cdot u = 0$ we can choose $y \in \mathcal{Y}$ arbitrarily.)
- Predict $\hat{z}_t = y \cdot x_t$ and then find out the true z_t.
- Update $s_{t+1} \leftarrow s_t + z_t x_t - z_t(x_t \cdot y_t)u$.

To apply Theorem 3 to our algorithm we need to evaluate the constants in our bounds. We are using the same hedging function as in external-regret matching, so the constants $A = 0$, $B = C = 1/2$, and $p = q = 2$ remain the same, as does the choice of the Euclidean norm. To determine M we need the longest vector in the augmented $\mathcal{Y}$. This vector has length 1.5: the size of the unaugmented $\mathcal{Y}$ is $\sqrt{5}/2$, and adding a constant component of 1 yields vectors of length up to $\sqrt{1 + 5/4} = 1.5$. For D we need the squared length of the largest possible update to s_t. Since $z_t x_t$ has length at most $\sqrt{5}/2$ and $z_t(x_t \cdot y_t) \in [-1, 1]$, the update has length at most 1.5, and we can take $D = 2.25$. Putting all of these values together, our final bound is

$$E(\rho_{t+1}) \leq 2.25\sqrt{t}$$

8.3 Other applications

A large variety of online prediction problems can be cast in our framework. These problems include online convex programming [1, 4, 5], p-norm perceptrons [2], path planning when costs are chosen by an adversary [10], planning in a Markov decision process when costs are chosen by an adversary [14], online pruning of a decision tree [15], and online balancing of a binary search tree [4]. In each case the bounds provided by the LH algorithm will be polynomial in the dimensionality of the appropriate hypothesis set and sublinear in the number of trials. Rather than re-proving all of the above results in our framework, we will illustrate the flexibility of the LH algorithm by turning now to a learning problem which has not previously been addressed in the literature: how to learn to play an extensive-form game.

9 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 [16], we can design algorithms which learn behavior strategies against unknown opponents, achieve $O(\sqrt{t})$ regret over t trials, and run in polynomial time. 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 by Freund and Schapire [11]; our work is the first to demonstrate this property in general extensive-form games.

We assume that our algorithm finds out, after each trial, both its cost $y_t \cdot c_t$ and the gradient of its cost c_t. Dealing with reduced feedback would be possible, but is beyond the scope of this paper. (For more information, see for example [17, 18].)

9.1 The sequence form

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 our feasible set $\mathcal{Y}$ to be the set $\mathcal{Y}^i_{\mathrm{seq}}$ of valid *sequence weight vectors* for our player. A sequence weight vector y for player i will contain one sequence weight $y^{s_i a_i}$ for each pair (s_i, a_i), where s_i is an information state where it is i's turn to move and a_i is one of i's available actions at s_i. All weights are nonnegative, and the probability of taking action a_i in state s_i is proportional to $y^{s_i a_i}$. The set $\mathcal{Y}$ is convex, and the payoff for a strategy $y \in \mathcal{Y}$ is a linear function of y when we hold the strategies of the other players fixed.

In more detail, we can represent player i's information state just before her kth move by a sequence of alternating observations and actions, ending in an observation:

$$s^i = (z^i_1, a^i_1, z^i_2, a^i_2, \ldots, z^i_k)$$

An edge x in the game tree is uniquely identified by the most recent sequences and actions for all players, $x = (s^1, a^1, s^2, a^2, \ldots)$.

Player i's policy can be represented by a weight $y^{s^i a^i}$ for each of her state-action pairs (s^i, a^i), defined as

$$y^{s^i a^i} = P(a^i_1 \mid s^i_1) P(a^i_2 \mid s^i_2) \ldots P(a^i_k \mid s^i_k) \tag{15}$$

Here k is the length of s^i, and s^i_j is the subsequence of s^i ending with z^i_j, so for example $s^i_k = s^i$. We have written $P(a^i_j \mid s^i_j)$ for the probability that player i will choose action a^i_j after having observed s^i_j.

The valid sequence weight vectors satisfy a set of linear constraints: for any state s^i, the weights $y^{s^i a^i}$ for different actions a^i share all terms in the product (15) except for the last. So, if we sum these weights, we can factor out the first $k-1$ terms and use the fact that probabilities sum to 1 to get rid of the kth term. If $k = 1$, there was only one term in the product to begin with, so we have:

$$\sum_{a^i_1} y^{s^i_1 a^i_1} = 1 \tag{16}$$

On the other hand, for $k > 1$, the first $k-1$ terms in (15) are just a sequence weight from the $(k-1)$st move, so we have:

$$\sum_{a^i_k} y^{s^i_k a^i_k} = y^{s^i_{k-1} a^i_{k-1}}$$

Together with the requirement of nonnegativity, we will write these constraints as

$$\mathcal{Y}^i_{\mathrm{seq}} = \{ y \geq 0 \mid A^i_{\mathrm{seq}} y = b^i_{\mathrm{seq}} \}$$

for a matrix A^i_{seq} and vector b^i_{seq}. Note that the total number of nonzero entries in the matrices A^i_{seq} and b^i_{seq} for all i is linear in the size of the original game tree. Also note that any vector $y \in \mathcal{Y}^i_{\text{seq}}$ corresponds to a valid strategy for player i: the probability of choosing action a given history s^i is

$$P(a \mid s^i) = y^{s^i a} \Big/ \sum_{a^i_k} y^{s^i_k a^i_k}$$

To conclude this subsection we will show that a player's expected cost is linear in her sequence weights. Given an edge x in a two-player game tree, determined by the sequence-action pairs (s^1, a^1) and (s^2, a^2) which the players must play to reach x, the probability of getting to x is just the product of the conditional probabilities of all of the actions required to reach x:

$$P(x) \quad = \quad P(a^1_1 \mid s^1_1) P(a^2_1 \mid s^2_1) P(a^1_2 \mid s^1_2) P(a^2_2 \mid s^2_2) \ldots$$

If we group together all of the terms for player 1's actions, we get a sequence weight for player 1, and similarly for player 2:

$$\begin{aligned} P(x) &= \left[P(a^1_1 \mid s^1_1) P(a^1_2 \mid s^1_2) \ldots\right] \left[P(a^2_1 \mid s^2_1) P(a^1_2 \mid s^2_2) \ldots\right] \\ &= y^{s^1 a^1} y^{s^2 a^2} \end{aligned}$$

Similarly, in an n-player game, the probability of reaching the edge

$$x = (s^1, a^1, s^2, a^2, \ldots, s^n, a^n)$$

is

$$P(x) = y^{s^1 a^1} y^{s^2 a^2} \ldots y^{s^n a^n}$$

If the cost to player i for traversing edge x is c^i_x, then i's total expected cost is

$$\sum_{x \in \text{edges}} c^i_x \, P(x) \; = \sum_{x = (s^1, a^1, \ldots, s^n, a^n) \in \text{edges}} c^i_x \, y^{s^1 a^1} y^{s^2 a^2} \ldots y^{s^n a^n} \tag{17}$$

which is linear in player i's sequence weights if we hold the weights for the other players fixed.

9.2 Algorithms

As noted above, if we are controlling player i, our algorithms will choose strategies $y \in \mathcal{Y} = \mathcal{Y}^i_{\text{seq}}$. They will receive, after each turn, a vector c_t which is the gradient with respect to y

of the expected total cost to player i. (We can compute c_t easily by differentiating (17).) The algorithms will then update their regret vector

$$s_t = u \sum_{i=1}^{t-1} y_t \cdot c_t - \sum_{i=1}^{t-1} c_t \tag{18}$$

Here u is a vector with $u \cdot y = 1$ for all $y \in \mathcal{Y}^i_{\text{seq}}$. For example, u can be zero everywhere except for 1s in the components s, a corresponding to some initial state s and all actions a. (Equation (16) guarantees that this choice of u satisfies $u \cdot y = 1$.)

Given s_t, our algorithms will choose y_t by an optimization involving $\mathcal{Y}^i_{\text{seq}}$, s_t, and a hedging function W. We can specify different no-regret algorithms by choosing various hedging functions. Good choices include quadratic and entropy-based hedging functions; these result in extensive-form versions of the external-regret matching and Hedge algorithms.

For example, the EF external-regret matching algorithm runs as follows: given the regret vector s_t from (18), solve the optimization problem

$$\bar{y} = \arg \max_{\bar{y} \in \bar{\mathcal{Y}}^i_{\text{seq}}} (s_t \cdot \bar{y} - \|\bar{y}\|_2^2/2) \tag{19}$$

and normalize $\bar{y}$ to get a feasible sequence weight vector $y_t \in \mathcal{Y}^i_{\text{seq}}$. The set $\bar{\mathcal{Y}}^i_{\text{seq}}$ can be written

$$\bar{\mathcal{Y}}^i_{\text{seq}} = \{\, y \geq 0 \mid A^i_{\text{seq}} y = \lambda b^i_{\text{seq}},\ \lambda \geq 0 \,\}$$

Since $\bar{\mathcal{Y}}^i_{\text{seq}}$ can be described by linear equalities and inequalities, the optimization problem (19) is a convex quadratic program and can be solved in polynomial time [19].

The EF Hedge algorithm solves instead the optimization problem

$$\bar{y} = \arg \max_{y \in \bar{\mathcal{Y}}^i_{\text{seq}}} (s_t \cdot y - W_1(y))$$

where W_1 is defined in Equation (5). Equivalently, we can solve

$$\begin{array}{rl} \text{maximize} & z \\ \text{subject to} & z \leq s_t \cdot y - \sum_i y_i \ln y_i \\ & \sum_i y_i = 1 \\ & y \in \bar{\mathcal{Y}}^i_{\text{seq}} \end{array} \tag{20}$$

Because this optimization problem is convex, with a polynomial number of linear constraints and a single simple nonlinear constraint, we can use a number of algorithms to solve it efficiently starting from a feasible point y^0. (We can get such a y^0 by, *e.g.*, renormalizing the sequence weights for the strategy which chooses actions uniformly at random.) For example, there is a fast separation oracle for the constraints in (20), so we can find a near-optimal y in polynomial time using the ellipsoid algorithm. Or, for better practical performance, we could use a log-barrier algorithm such as the one described in Boyd and Vandenberghe's text [19].

9.3 Regret bounds

By evaluating the constants in Theorem 3 we can show regret bounds for the extensive-form algorithms. The bound for extensive-form external-regret matching is

$$E(\rho_{t+1}(y)) \leq d\sqrt{td} \tag{21}$$

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

$$E(\rho_{t+1}(y)) \leq 2d\sqrt{2t \ln d} \tag{22}$$

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

In more detail, the only change in regret bounds when we move from the original Hedge and external-regret matching algorithms to their extensive-form versions is that, since we have changed the hypothesis space from the probability simplex to the more complicated set $\mathcal{Y}^i_{\text{seq}}$, the constants D and M are different.

For the quadratic hedging function, the constants $A = 0$, $B = C = 1/2$, and $p = q = 2$ remain unchanged from the analysis of the original external-regret matching algorithm. M is the size of a 2-norm ball enclosing $\mathcal{Y}^i_{\text{seq}}$. This constant depends on exactly which game we are playing, but it is bounded by the dimension d of the sequence weight vector since each sequence weight is in $[0, 1]$.

The bound D on the size of the regret update depends similarly on exactly which game we are playing. We will we assume that the individual edge costs are in $[0, 1]$ and that the total cost along any path is no more than 1. The first assumption means that our cost vector c_t is in $[0, 1]^d$: according to (17), a sequence weight $y^{s_i a_i}$ affects the total cost only through terms which correspond to the game tree edges that are consistent with player i playing the actions specified in s_i and a_i. The weight of $y^{s_i a_i}$ in each of these terms is the product of the cost of the corresponding edge with the conditional probability that we will reach the edge given that player i plays her prescribed actions and the other players follow their given policies. Since these conditional probabilities sum to no more than 1 and since the costs are in $[0, 1]$, the gradient with respect to $y^{s_i a_i}$ will be in $[0, 1]$. Finally, u is in $[0, 1]^d$ and $y_t \cdot c_t \in [0, 1]$, so the regret update is in $[-1, 1]^d$. The 2-norm radius of $[-1, 1]^d$ is d, so we can take $D = d$. Applying Theorem 3 to the above set of constants yields the bound in Equation (21).

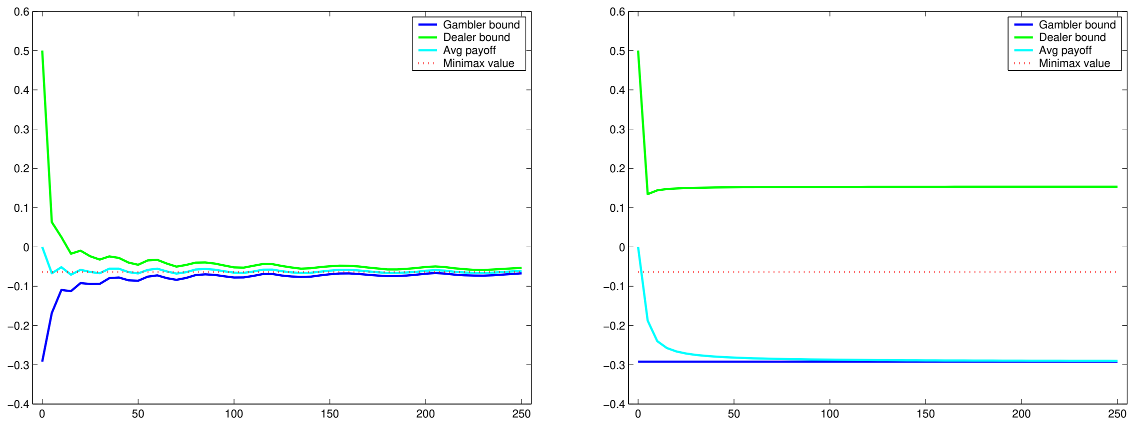

Figure 6: Performance in self-play (left) and against a fixed opponent (right).

For the entropy hedging function, M is the size of a 1-norm ball enclosing $\mathcal{Y}$, so we can take $M = d$. And, D is the size of a max-norm ball enclosing our regret updates, which is $D = 1$. The constants $A = \ln d$, $B = 1$, $C = 1/2$, $p = 1$, and $q = \infty$ remain unchanged from ordinary Hedge. Applying Theorem 3 to the above set of constants yields the bound in Equation (22).

10 Experiments

To demonstrate that our theoretical bounds translate to good practical performance, we implemented the extensive-form external-regret matching algorithm of Section 9 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 bring her bet up to \$1. If either player bets \$0 when the other has already bet \$1, that player folds and loses her 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. The biggest strategic difference between one-card poker and larger variants such as draw, stud, or hold-em

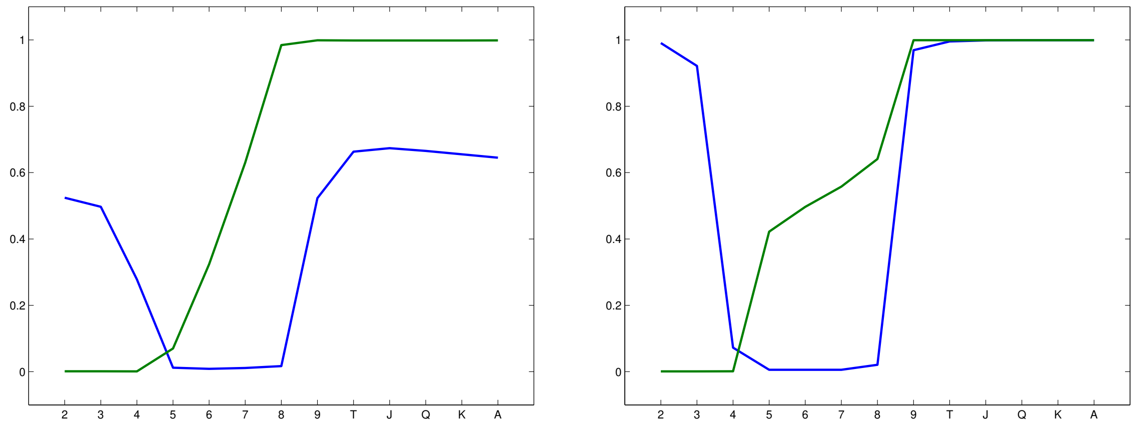

Figure 7: Minimax one-card poker strategies learned by self-play. Left: gambler bet probabilities holding different cards. First round in blue, second round in green. Right: dealer bet probabilities. Probability after hearing gambler pass in blue, after hearing gambler bet in green.

is the idea of hand potential: while 45679 and 24679 are almost equally strong hands in a showdown (they are both 9-high), holding 45679 early in the game is much more valuable because replacing the 9 with either a 3 or an 8 turns it into a straight.

Figure 6 shows the results of two typical runs: in both panels the dealer is using our no-regret algorithm. In the left panel the gambler is also using our no-regret algorithm, while in the right panel the gambler is playing a fixed policy. The x-axis shows number of hands played; the y-axis shows the average payoff per hand from the dealer to the gambler. The value of the game, $-\$0.064$, 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, she 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, as shown in Figure 7. (Note that there are many minimax strategies for one-card poker, so these plots are different from the ones reported in, *e.g.*, [16].)

The curves in the left panel of Figure 6 show an interesting effect: the small, damping oscillations result from the dealer and the gambler "chasing" each other around a minimax strategy. One player will learn to exploit a weakness in the other, but in doing so will open up a weakness in her own play; then the second player will adapt to try to take advantage of the first, and the cycle will repeat. Each weakness will be smaller than the last, so the sequence of strategies will converge to a minimax equilibrium. This cycling behavior is a common phenomenon when two learning players play against each other. Many learning algorithms will cycle so strongly that they fail to achieve the value of the game, but our regret bounds eliminate this possibility.

11 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 predictions. We have proved regret bounds for LH algorithms and demonstrated experimentally that these bounds lead to good predictive performance in practice. The regret bounds for LH algorithms have low-order dependences on d, the number of dimensions in the hypothesis set $\mathcal{Y}$. This low-order dependence means that the LH algorithms can learn well in prediction problems with complicated hypothesis sets; these problems would otherwise require an impractical amount of training data and computation time.

Our work builds on previous work in online learning and online convex programming. Our contributions include a new, deterministic algorithm; a simple, general proof; 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, which allows us to construct potential functions for arbitrary convex hypothesis sets. 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).

Acknowledgments

Thanks to Amy Greenwald, Martin Zinkevich, and Sebastian Thrun for many helpful discussions during the course of this work. Thanks also to Yoav Shoham and his research group, who listened to the first presentation of this work in October, 2003 and provided many helpful comments then and thereafter. This work was supported by NSF grant EF–0331657, DARPA contract F30602–01–C–0219, and DARPA contract NBCH–1020014. The opinions and conclusions are the author's and do not reflect those of the US government or its agencies.

References

[1] Geoffrey J. Gordon. *Approximate Solutions to Markov Decision Processes*. PhD thesis, Carnegie Mellon University, 1999.

[2] Nicolò Cesa-Bianchi and Gábor Lugosi. Potential-based algorithms in on-line prediction and game theory. *Machine Learning*, 51:239–261, 2003.

[3] David Blackwell. An analogue of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 6(1):1–8, 1956.

[4] Adam Kalai and Santosh Vempala. Geometric algorithms for online optimization. Technical Report MIT-LCS-TR-861, MIT, 2002.

[5] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *20th ICML*, 2003.

[6] James F. Hannan. Approximation to Bayes risk in repeated play. In M. Dresher, A. Tucker, and P. Wolfe, editors, *Contributions to the Theory of Games*, volume 3, pages 97–139. Princeton University Press, 1957.

[7] Geoffrey J. Gordon. Regret bounds for prediction problems. In *Proceedings of the ACM Conference on Computational Learning Theory*, 1999.

[8] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. Technical Report CMU-CS-03-110, Carnegie Mellon School of Computer Science, 2003.

[9] David P. Helmbold and Robert E. Schapire. Predicting nearly as well as the best pruning of a decision tree. In *Proceedings of COLT*, pages 61–68, 1995.

[10] Eiji Takimoto and Manfred Warmuth. Path kernels and multiplicative updates. In *COLT*, 2002.

[11] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT 95*, pages 23–37. Springer-Verlag, 1995.

[12] Nick Littlestone and Manfred Warmuth. The weighted majority algorithm. Technical Report UCSC-CRL-91-28, University of California Santa Cruz, 1992.

[13] Sergiu Hart and Andreu Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.

[14] H. Brendan McMahan, Geoffrey J. Gordon, and Avrim Blum. Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

[15] David P. Helmbold and Robert E. Schapire. Predicting nearly as well as the best pruning of a decision tree. In *COLT*, 1995.

[16] D. Koller, N. Meggido, and B. von Stengel. Efficient computation of equilibria for extensive two-person games. *Games and Economic Behaviour*, 14(2), 1996.

[17] Abraham Flaxman, Adam Kalai, and H. Brendan McMahan. Online convex optimization in the bandit setting: Gradient descent without a gradient. In *Symposium on Discrete Algorithms (SODA)*, 2005.

[18] Robert Kleinberg. Nearly tight bounds for the continuum-armed bandit problem. In *Advances in Neural Information Processing Systems*, volume 18, 2005.

[19] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[20] R. Tyrell Rockafellar. *Convex Analysis*. Princeton University Press, New Jersey, 1970.

[21] Jonathan M. Borwein and Adrian S. Lewis. *Convex Analysis and Nonlinear Optimization: Theory and Examples*. Springer-Verlag, New York, 2000.

A Proof of main results—I

This appendix contains the proof of Theorem 3. The result as given in Section 7 is a straightforward combination of Theorems 7 and 8, stated and proved below.

Our proof proceeds in three steps: first we will prove a general result about gradient descent (Theorem 5 below) which uses our upper bound on F, together with the assumption that $E(s_{t+1}-s_t)$ never points in the same direction as 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 bound on F to show that s_t itself 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 (9) on p. 10 holds for all s and Δ.[8]

We also need a condition on our updates to s_t: we need them never to point in the same direction as the gradient of $F(s_t)$. That is, we need

$$E((s_{t+1}-s_t)\cdot f(s_t) \mid s_t) \leq 0 \tag{23}$$

We will call Equation (23) the *generalized Blackwell condition* since it is similar to one of the conditions of Blackwell's approachability theorem [3]. Our first theorem proves a general bound on the growth rate of $F(s_t)$ using conditions (9) and (23).

Theorem 5 (Gradient descent) *Let $F(s)$ and $f(s)$ satisfy Equation (9) 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 that, for all t, $E(x_t \cdot f(s_t) \mid s_t) \leq 0$. 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

$$F(s_1) - F(s_1) \;\leq\; 0$$

For $t \geq 1$, assume that

$$E(F(s_t) \mid s_1) \leq F(s_1) + (t-1)CD$$

[8] The text around Equation (9) specifies that F is convex and that $\|\cdot\|$ is a finite norm, but Theorem 5 holds in the more general case when F may be non-convex and $\|s\|$ may be ∞ or 0. If $\|\cdot\|$ is a norm and F is convex (as will be the case in our application of Theorem 5 below), then Equation (9) implies that F is differentiable everywhere and that f is its gradient.

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 (9); the third, from taking $E(\cdot \mid s_t)$ on both sides, then 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. □

A.2 The expected change in s_t

We would like to apply Theorem 5 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 vectors s_t that satisfies the necessary assumptions. We have already assumed, in Equation (12), 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 (23). The following lemma does so:

Lemma 6 *The Lagrangian Hedging algorithm produces a sequence of regret vectors 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 the variable $\bar{y}_t$ from Figure 2. This choice means that the variable y_t from Figure 2 satisfies $ky_t = f_t$ where $k = (\bar{y}_t \cdot u_t) \geq 0$: in the **then** clause of Figure 2 we have $\bar{y}_t \cdot u > 0$ so we can just multiply through. In the **else** clause, $\bar{y}_t \cdot u = 0$. This means $\bar{y}_t = 0$: since $\bar{y}_t \in \bar{\mathcal{Y}}$, we can write $\bar{y}_t = \lambda y$ for some $y \in \mathcal{Y}$ and $\lambda \geq 0$. Dotting with u gives us

$$u \cdot \bar{y}_t = \lambda u \cdot y$$

or

$$0 = \lambda$$

since $u \cdot y = 1$ for any $y \in \mathcal{Y}$ by the definition of u. So, $\bar{y}_t = 0 = ky_t$.

Now, Equation (1) tells us that the expected change in the regret vector is

$$E(s_{t+1} - s_t \mid s_t) = (c_t \cdot y_t)u - c_t$$

where c_t is chosen by the opponent but must be independent of y_t. Taking the dot product with y_t yields

$$E((s_{t+1} - s_t) \cdot y_t \mid s_t) = (c_t \cdot y_t)(u \cdot y_t) - c_t \cdot y_t = 0$$

since $u \cdot y_t = 1$. Note that this expected value does not depend on c_t: the opponent can't influence the expected component of $s_{t+1} - s_t$ along y_t. Multiplying both sides by k and using the identity $ky_t = f_t$ inside the expectation, we have

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

which proves the desired result. □

A.3 Bounds on the gradient form

In addition to the upper bounds in Equation (9), 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. (Adjusting our learning rate, as described below in Section A.4, will allow us to improve some of these bounds.)

To collect all of these results into a single theorem, we will parameterize our lower bound on F by an exponent $1 \le p \le 2$, as shown in Equation (10) on p. 10. To make (10) be a non-vacuous lower bound, we will require $\|\cdot\|$ to be a norm rather than a seminorm. (That is, we will require $(\|x\| = 0) \Leftrightarrow (x = 0)$. Note that $\|\cdot\|$ must be finite since F is finite.) With our lower bound we have the following theorem:

Theorem 7 *Suppose the potential function F is convex and satisfies Equations (4), (9), and (10) for constants A, B, C and p and a norm $\|\cdot\|$. Suppose that the problem definition is bounded according to (11) and (12) for constants M and D. Then the LH algorithm (Figure 2) achieves expected regret*

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

versus any hypothesis $y \in \mathcal{Y}$.

Proof: Equations (9) and (12) together with Lemma 6 show that F, f, and the update $s_{t+1} - s_t$ satisfy the assumptions of Theorem 5. So,

$$E(F(s_{t+1}) \mid s_1) - F(s_1) \le tCD$$

Since s_1 is a fixed constant we can discard the conditioning, and since $s_1 \in \mathcal{S}$ we have $F(s_1) \le 0$ by Equation (4). So,

$$E(F(s_{t+1})) \le tCD$$

Since F is convex, Jensen's inequality tells us that $F(E(s_{t+1})) \le E(F(s_{t+1}))$. So, writing $\bar{s} = E(s_{t+1})$, we have

$$F(\bar{s}) \le tCD$$

Adding A on both sides and using the fact that $tCD + A \ge 0$, we also have

$$[F(\bar{s}) + A]_+ \le tCD + A$$

Now, applying (10) shows that

$$B \inf_{s \in \mathcal{S}} \|\bar{s} - s\|^p \le tCD + A \tag{24}$$

The function $x^{1/p}$ is monotone on $\mathbb{R}^+$; so, we can apply it to both sides of Equation (24) and then move it inside the inf operator on the left-hand side:

$$B^{1/p} \inf_{s \in \mathcal{S}} \|\bar{s} - s\| \le (tCD + A)^{1/p} \tag{25}$$

Now pick any $y \in \mathcal{Y}$ and $s \in \mathcal{S}$. Our expected regret versus y is

$$E(\rho_{t+1}(y)) = \bar{s} \cdot y \le (\bar{s} - s) \cdot y$$

since $s \cdot y \le 0$. So, for any $y \in \mathcal{Y}$ and $s \in \mathcal{S}$,

$$E(\rho_{t+1}(y)) \le (\bar{s} - s) \cdot y \le \|\bar{s} - s\| \, \|y\|_\circ \le M\|\bar{s} - s\| \tag{26}$$

by Hölder's inequality and bound (11). Since $s \in \mathcal{S}$ was arbitrary, we will pick the s which makes our bound tightest:

$$E(\rho_{t+1}(y)) \le M \inf_{s \in \mathcal{S}} \|\bar{s} - s\|$$

Finally, substituting in Equation (25) gives us

$$E(\rho_{t+1}(y)) \le M((tCD + A)/B)^{1/p}$$

which is the desired result. $\square$

A.4 Adjusting the learning rate

Theorem 7 shows that the LH algorithm is no-regret so long as $p > 1$. Some algorithms (for example, weighted majority) need $p = 1$ in their analysis; when $p = 1$, we can use the standard trick of an adjustable learning rate, together with prior knowledge of the number of trials, to achieve regret which is sublinear in t. For generality we will calculate the effect of adjusting the learning rate for $1 \le p < 2$, although in practice the $p = 1$ case is the most important.

As described in Section 5, we can add a learning rate η to the LH algorithm by replacing $F(s)$ with $G(s) = F(\eta s)$. If F satisfies Equations (9) and (10) with constants A, B, C, and p, then G satisfies them as well but with different constants: since $\partial G(s) = \eta \partial F(\eta s)$,

$$\begin{aligned} G(s+x) &= F(\eta s + \eta x) \\ &\le F(\eta s) + \eta x \cdot f(\eta s) + C\|\eta x\|^2 \\ &\le G(s) + x \cdot g(s) + \eta^2 C\|x\|^2 \end{aligned}$$

And, since $\eta s' \in \mathcal{S} \Leftrightarrow s' \in \mathcal{S}$,

$$\begin{aligned} [G(s) + A]_+ &= [F(\eta s) + A]_+ \\ &\ge \inf_{s' \in \mathcal{S}} B\|\eta s - s'\|^p \\ &= \inf_{s' \in \mathcal{S}} B\|\eta s - \eta s'\|^p \\ &= \inf_{s' \in \mathcal{S}} \eta^p B\|s - s'\|^p \end{aligned}$$

So, using a learning rate η changes the constants for Equations (9) and (10) according to $A \mapsto A$, $B \mapsto \eta^p B$, $C \mapsto \eta^2 C$, and $p \mapsto p$. By setting η to optimize these constants we can now prove the following theorem:

Theorem 8 *Suppose that F is convex and satisfies Equations (9) and (10) with constants A, B, C, and $1 \le p < 2$ and the norm $\|\cdot\|$. Suppose our problem definition has constants M and D in Equations (11) and (12). Let t be the anticipated number of trials, and define $G(s) = F(\eta s)$, where*

$$\eta = \sqrt{pA/(tCD(2-p))}$$

Then the LH algorithm with potential G achieves regret $O(\sqrt{t})$. In particular, if $p = 1$, we have

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

and

$$E(\rho_{t+1}(y)) \le (2M/B)\sqrt{tACD}$$

for any hypothesis $y \in \mathcal{Y}$.

Proof: Theorem 7 shows

$$E(\rho_{t+1}(y)) \leq M((t\eta^2 CD + A)/(\eta^p B))^{1/p}$$

or equivalently

$$E(\rho_{t+1}(y)) \leq M((t\eta^{2-p} CD + A\eta^{-p})/B)^{1/p} \tag{27}$$

Minimizing the above bound with respect to η is equivalent to solving

$$\frac{d}{d\eta}\left[t\eta^{2-p}CD + A\eta^{-p}\right] = 0$$

Since $0 < p < 2$, differentiating yields

$$(2-p)t\eta^{1-p}CD = pA\eta^{-p-1}$$

and therefore

$$\eta^2 = pA/(tCD(2-p))$$

which is the learning rate given in the theorem. Substituting this value of η back into our bound gives

$$\begin{aligned} E(\rho_{t+1}(y)) &\leq M((t\eta^2 CD + A)/B)^{1/p}/\eta \\ &= M((pA/(2-p) + A)/B)^{1/p}\sqrt{tCD(2-p)/(pA)} \\ &= O(\sqrt{t}) \end{aligned}$$

as required. When $p = 1$, the learning rate simplifies to $\sqrt{A/(tCD)}$ and the regret bound simplifies to $(2M/B)\sqrt{tACD}$. □

Note that in order to achieve sublinear regret for $p = 1$ we needed advance knowledge of the number of trials.[9] This sort of dependence on p is typical of results in the literature: when our potential function is superlinear the algorithm can in effect choose its own learning rate, while if the potential is merely linear in some direction leading away from $\mathcal{S}$ we need to select a learning rate based on external knowledge.

As $A \downarrow 0$, the recommended learning rate gets smaller and smaller. If A were 0 the recommendation would be $\eta = 0$, which seems like a contradiction. But, it is not possible to have $p < 2$ and $A = 0$: take $\lambda > 0$ and $\Delta \notin \mathcal{S}$ with $f(0) \cdot \Delta \leq 0$. (Since $f(0) \in \bar{\mathcal{Y}}$, such a Δ always exists: $\mathcal{S}$ is contained in any halfspace whose normal is in $\bar{\mathcal{Y}}$, and since we have assumed $\mathcal{Y}$ has at least two distinct elements the containment must be strict.) Then Equation (9) at regret vector $s = 0$ and increment $\lambda\Delta$ requires

$$F(\lambda\Delta) \leq F(0) + \lambda\Delta \cdot f(0) + C\|\lambda\Delta\|^2 \leq C\|\lambda\Delta\|^2$$

[9] At the cost of some complexity we could have used a decreasing sequence of learning rates to sidestep this requirement.

since $F(0) \leq 0$ (because $0 \in \mathcal{S}$). And, Equation (10) requires

$$F(\lambda\Delta) \geq B \inf_{s' \in \mathcal{S}} \|\lambda\Delta - s'\|^p$$

These two bounds are inconsistent: combined, they require

$$\lambda^2 \cdot \text{constant} \geq \lambda^p \cdot \text{constant}$$

with both constants strictly positive, which cannot hold as $\lambda \downarrow 0$ since $p < 2$.

As $p \uparrow 2$, the recommended learning rate gets larger and larger. If $p = 2$, the recommended learning rate will be $\eta = \infty$ (unless $A = 0$, in which case Equation (27) is independent of η): while the analysis in the proof of Theorem 8 doesn't apply, it is easy to see that increasing η doesn't alter the ratio C/B in Equation (27) and decreases A/B, thereby improving the bound. In practice, if p is near or equal to 2 and $A > 0$, we would recommend setting η as large as is practical.

B Proof of main results—II

Theorems 7 and 8 bound the regret of the gradient form of the LH algorithm in terms of properties of F. For the optimization form we are not given the potential function F directly, so we cannot check the conditions of these theorems. Instead we define F in terms of the hedging function W using Equation 7. Unlike F, there is no need for W to be differentiable, so long as it satisfies the required assumptions.

In this section we describe 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. The ability to transfer bounds means that, when we analyze or implement the optimization form of the LH algorithm, we never have to evaluate the potential function F or its derivative explicitly.

Our bounds on W are detailed above, in Section 7. With these bounds on W, we can prove the required bounds on the potential function F:

Theorem 9 *Suppose that the hedging function W is closed, convex, nonnegative, and satisfies Equations (13) and (14) with the constants A, B, C, and $2 \leq q \leq \infty$ and the finite norm $\|\cdot\|_\circ$. Suppose the set $\bar{\mathcal{Y}} \cap \operatorname{rel\,int} \operatorname{dom} W$ is nonempty. Define p so that $\frac{1}{p} + \frac{1}{q} = 1$. Then the function F defined by Equation (7) is closed and convex and satisfies Equations (4), (9), and (10) with constants A, B, C, and p and norm $\|\cdot\|$.*

Since W and related functions may not be differentiable, we will use the notation of convex analysis to prove our bounds; see Appendix E for definitions. In this notation Equation (7) is equivalent to

$$F = (I_{\bar{\mathcal{Y}}} + W)^* \tag{28}$$

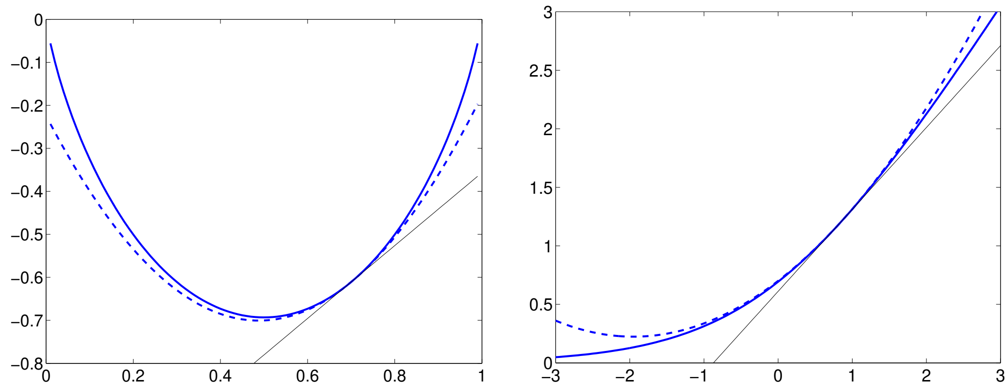

Figure 8: Illustration of how to transfer bounds between a function and its dual. On the left, the negentropy function and a quadratic lower bound; on the right, $\ln(1 + e^x)$ and the dual quadratic upper bound.

Here $I_{\bar{\mathcal{Y}}}$ represents the feasible region of the optimization in (7), while W is the nonlinear part of the objective. (The linear part of the objective corresponds to the argument of F.) By moving the duality operator inside the parentheses, we can see that Equation (28) is also equivalent to

$$F = I_{\mathcal{S}} \,\Box\, W^* \tag{29}$$

since infimal convolution is dual to addition and $I_{\mathcal{S}}$ is dual to $I_{\bar{\mathcal{Y}}}$.

Our bounds on F follow from the simple observation that the duality operator reverses inequalities between functions, as illustrated in Figure 8: for closed convex functions F and G, if $F^*(y) \geq G^*(y)$ for all y, then $G(s) \geq F(s)$ for all s. This fact is a direct consequence of the definition of duality:

$$G(s) = \sup_y \left[s \cdot y - G^*(y)\right] \geq \sup_y \left[s \cdot y - F^*(y)\right] = F(s) \tag{30}$$

where the inequality holds because substituting F^* for G^* reduces the expression in square brackets at every value of y, and therefore reduces the supremum.

We can use the inequality (30) almost directly to turn our upper bound on W into a lower bound on F: all we will need to do in our proof below is add $I_{\bar{\mathcal{Y}}}$ to both sides of Equation (14) and take the dual. To prove our upper bound on F, on the other hand, requires a slightly more complicated argument.

Returning to Figure 8, notice that the bound on the left is tangent to $F^*(y)$ at the input $y_0 = 0.7$ with slope $s_0 \approx 0.85$, while the dual bound on the right is tangent at to $F(s)$ at the input s_0 with slope y_0. This sort of correspondence holds in general: the slope of a function translates to the argument of its dual, and vice versa. So, if we start with a lower bound on F^* of the form we could imagine deriving from Equation (13)

$$F^*(y) \geq F^*_{\text{bound}}(y) = F^*(y_0) + (y - y_0) \cdot s_0 + L\|y - y_0\|_\circ^2/2 \tag{31}$$

which is tangent at $y = y_0$ with slope s_0, we end up with an upper bound on $F(s)$ which is tangent at $s = s_0$. To prove (9), we need to produce bounds on F which are tangent at every possible input s_0; so, we need to start from bounds on F^* which have every possible slope s_0 at their tangent points. The proof below demonstrates how to construct such bounds from Equation (13).

Proof **(of Theorem 9)**: It is immediate that F is closed and convex, since F is defined as the dual of another function and the output of the duality operator is always closed and convex. Equation (4) is also immediate: in (7),

$$s \cdot \bar{y} - W(\bar{y}) \leq s \cdot \bar{y}$$

since $W(y) \geq 0$; so, since $s \cdot \bar{y} \leq 0$ for all $s \in \mathcal{S}$ and $\bar{y} \in \bar{\mathcal{Y}}$, $F(s) \leq 0$ for all $s \in \mathcal{S}$.

Let us now prove the lower bound on F, Equation (10). We have assumed (Equation (14)) that

$$\text{conv}\ \min(W(y) - A + I_{\bar{\mathcal{Y}}}(y),\, I_0(y)) \leq B\|y/B\|_\circ^q \qquad \forall y \in \bar{\mathcal{Y}}$$

Adding $I_{\bar{\mathcal{Y}}}$ to both sides yields

$$\text{conv}\ \min(W(y) - A + I_{\bar{\mathcal{Y}}}(y),\, I_0(y)) \leq B\|y/B\|_\circ^q + I_{\bar{\mathcal{Y}}}(y) \tag{32}$$

The left-hand side was already infinite for $y \notin \bar{\mathcal{Y}}$, so adding $I_{\bar{\mathcal{Y}}}$ had no effect. Note that we have dropped the qualifier $\forall y \in \bar{\mathcal{Y}}$ since (32) is clearly true if $y \notin \bar{\mathcal{Y}}$.

We will next take duals on both sides of (32). For any two functions X and Y, the dual of conv $\min(X, Y)$ is $\max(X^*, Y^*)$ and the dual of $X + Y$ is $X^* \mathbin{\square} Y^*$. The dual of the indicator function for a cone is the indicator function for the dual cone; for example, the dual of I_0 is $I_{\mathbb{R}^d} = 0$. So, writing s for the dual variable, we have

$$\max((W(y) - A + I_{\bar{\mathcal{Y}}}(y))^*(s),\, 0) \geq (B\|y/B\|_\circ^q)^*(s) \mathbin{\square} I_{\mathcal{S}}(s)$$

Since $F^* = W + I_{\bar{\mathcal{Y}}}$, we can simplify the first argument of the max:

$$\max(F(s) + A,\, 0) \geq (B\|y/B\|_\circ^q)^*(s) \mathbin{\square} I_{\mathcal{S}}(s)$$

The dual of $\|\cdot\|_\circ^q$ is $\|\cdot\|^p$, and for any function X the dual of $aX(y)$ is $aX^*(s/a)$, so the dual of $B\|y/B\|_\circ^q$ is $B\|s\|^p$. That gives us

$$\max(F(s) + A,\, 0) \geq B\|s\|^p \mathbin{\square} I_{\mathcal{S}}(s)$$

which is equivalent to Equation (10) as desired.

For the upper bound on F, Equation (9), we can use some simple identities to compute the dual of a function of the form F^*_{bound} given in (31): first, the dual of any multiple of a squared norm is a multiple of the squared dual norm.

$$(L\|\cdot\|^2/2)^* = (1/L)\|\cdot\|_\circ^2/2 \tag{33}$$

Second, adding a linear function to an arbitrary convex function G just shifts the dual of G without changing its basic shape:

$$(a \cdot s + b + G(s))^* = G^*(y - a) - b \tag{34}$$

Finally, if we have a point $(y_0, G^*(y_0))$ where there is a tangent to G^* of slope s_0, then the function $G^*(y) - y \cdot s_0$ has a tangent of slope 0 at $y = y_0$. So, y_0 is a minimum of $G^*(y) - y \cdot s_0$, and

$$G(s_0) = \sup_y (y \cdot s_0 - G^*(y)) = y_0 \cdot s_0 - G^*(y_0) \tag{35}$$

Combining the identities (33) and (34), the dual of

$$L\|y\|_\circ^2/2 + s_0 \cdot y + F^*(y_0)$$

is

$$(1/L)\|s - s_0\|^2/2 - F^*(y_0)$$

Using Equation (34) again (in the opposite direction) for the substitution $y \mapsto (y - y_0)$ tells us that the dual of F^*_{bound} is

$$F_{\text{bound}}(s) = (1/L)\|s - s_0\|^2/2 - F^*(y_0) + s \cdot y_0$$

Adding and subtracting $s_0 \cdot y_0$ and using (35) gives us

$$F_{\text{bound}}(s) = F(s_0) + (s - s_0) \cdot y_0 + (1/L)\|s - s_0\|^2/2 \tag{36}$$

As mentioned above in the main text, $F_{\text{bound}}(s_0) = F(s_0)$ and by Equation (30) we have $F_{\text{bound}}(s) \geq F(s)$ for all s. So, to prove our result we need to be able to construct an appropriate F^*_{bound} from Equation (13) for any desired slope s_0.

First we will show that F must be finite everywhere. We have assumed that there exists a point $y_0 \in \bar{\mathcal{Y}} \cap \operatorname{rel\,int}\operatorname{dom} W \subseteq \operatorname{dom}\partial W$. Write s_0 for an arbitrary element of $\partial W(y_0)$. Now Equation (13) tells us that

$$\begin{aligned}
F(s) &= \sup_{\bar{y} \in \bar{\mathcal{Y}}} (\bar{y} \cdot s - W(\bar{y})) \\
&\leq \sup_{\bar{y} \in \bar{\mathcal{Y}}} (\bar{y} \cdot s - W(y_0) - s_0 \cdot (\bar{y} - y_0) - (1/4C)\|\bar{y} - y_0\|_\circ^2) \\
&< \infty
\end{aligned}$$

because the expression inside the supremum is bounded above (along every line through y_0 it is concave and quadratic, with bounded slope at y_0).

Since F is finite everywhere, ∂F is nonempty everywhere. So, given a desired s_0, pick $y_0 \in \partial F(s_0)$; we will build an F^*_{bound} function of the form given in Equation (31) using this choice of y_0.

By duality we have $s_0 \in \partial F^*(y_0)$. Since $F^* = I_{\bar{\mathcal{Y}}} + W$ we have $s_0 = s_1 + s_2$ with $s_1 \in \partial I_{\bar{\mathcal{Y}}}(y_0)$ and $s_2 \in \partial W(y_0)$ by Theorem 23.8 of [20, p. 223]. Theorem 23.8 applies because we have assumed that $\bar{\mathcal{Y}} \cap \operatorname{rel\,int\,dom} W$ is nonempty.

The existence of s_1 tells us that $y_0 \in \bar{\mathcal{Y}}$, and similarly the existence of s_2 tells us that $y_0 \in \operatorname{dom} \partial W$. So by assumption Equation (13) holds for y_0 and s_2:

$$W(y) \geq W(y_0) + (y - y_0) \cdot s_2 + (1/4C)\|y - y_0\|_\circ^2 \qquad \forall y$$

And by definition of subgradient,

$$I_{\bar{\mathcal{Y}}}(y) \geq I(y_0) + (y - y_0) \cdot s_1 \qquad \forall y$$

Adding these two inequalities yields

$$F^*(y) \geq F^*(y_0) + (y - y_0) \cdot s_0 + (1/4C)\|y - y_0\|_\circ^2 \qquad \forall y \tag{37}$$

Picking $L = 1/2C$, we can identify Equation (37) with Equation (31). So, taking the dual of both sides, we have

$$F(s) \leq F(s_0) + (s - s_0) \cdot y_0 + C\|s - s_0\|^2 \qquad \forall s$$

as we derived in Equation (36). Since s_0 was arbitrary, we have now shown that F satisfies (9), which finishes the proof of our theorem. □

C Additional proofs

In this section we will prove that the two forms of the LH algorithm are well-defined and that the optimization form is a special case of the gradient form.

Proof **(of Theorem 1)**: Define $\bar{\mathcal{Y}}$ as in Equation (3). If we can show that $\bar{y}_t \in \bar{\mathcal{Y}}$ then we are done: if $\bar{y}_t = \lambda y$, then $\bar{y}_t \cdot u = \lambda$. Either $\lambda > 0$, in which case the **then** clause in Figure 2 will pick $y_t = y \in \mathcal{Y}$, or $\lambda = 0$, in which case the **else** clause will pick $y_t \in \mathcal{Y}$.

By convexity, since $\bar{y}_t \in \partial F(s_t)$,

$$F(s) \geq F(s_t) + (s - s_t) \cdot \bar{y}_t$$

For all $s \in \mathcal{S}$ we have $F(s) \leq 0$, so

$$0 \geq F(s_t) + (s - s_t) \cdot \bar{y}_t \qquad \forall s \in \mathcal{S}$$

or, rearranging terms,

$$s_t \cdot \bar{y}_t - F(s_t) \geq s \cdot \bar{y}_t \qquad \forall s \in \mathcal{S}$$

Since $\alpha s \in \mathcal{S}$ for all $\alpha > 0$, we also have

$$\begin{aligned} s_t \cdot \bar{y}_t - F(s_t) &\geq \alpha s \cdot \bar{y}_t \\ (s_t \cdot \bar{y}_t - F(s_t))/\alpha &\geq s \cdot \bar{y}_t \\ 0 &\geq s \cdot \bar{y}_t \end{aligned} \tag{38}$$

for all $s \in \mathcal{S}$, where the last line follows because we can make α arbitrarily large.

Now, $\mathcal{S}$ was defined as $\mathcal{Y}^\perp$, or equivalently $\bar{\mathcal{Y}}^\perp$. $\bar{\mathcal{Y}}$ is a closed convex cone, since $\mathcal{Y}$ is closed and convex; so, saying $\mathcal{S} = \bar{\mathcal{Y}}^\perp$ is equivalent to saying $\bar{\mathcal{Y}} = \mathcal{S}^\perp$. But, $\mathcal{S}^\perp$ is exactly the set of vectors y with $s \cdot y \leq 0$ for all $s \in \mathcal{S}$; so, inequality (38) shows that $\bar{y}_t \in \bar{\mathcal{Y}}$. □

PROOF **(of Theorem 2)**: To show $F(s) \leq 0$ for all $s \in \mathcal{S}$, recall that $s \cdot \bar{y} \leq 0$ for all $s \in \mathcal{S}$ and $\bar{y} \in \bar{\mathcal{Y}}$. Since $W(\bar{y}) \geq 0$, that means that both terms inside the supremum in (7) are nonpositive for all feasible $\bar{y}$ when $s \in \mathcal{S}$. Since there is at least one feasible $\bar{y}$, the value of the supremum must also be nonpositive.

To show equivalence, consider any $\bar{y}$ which achieves the supremum in (7). Such a $\bar{y}$ must exist, since $W(\bar{y}) + I_{\bar{\mathcal{Y}}}(\bar{y}) - s \cdot \bar{y}$ is closed, convex, not everywhere infinite, and has no directions of recession (see [20, Theorem 27.1(d), p. 265]). For this $\bar{y}$,

$$\begin{aligned} F(s + \Delta) &= \sup_{\bar{y}' \in \bar{\mathcal{Y}}} ((s + \Delta) \cdot \bar{y}' - W(\bar{y}')) \\ &\geq (s + \Delta) \cdot \bar{y} - W(\bar{y}) \\ &= \Delta \cdot \bar{y} + (s \cdot \bar{y} - W(\bar{y})) \\ &= \Delta \cdot \bar{y} + F(s) \end{aligned}$$

So, $\bar{y} \in \partial F(s)$, which is what was required. □

D Analysis of the entropy function

This section derives the constants required for using the entropy function in the bounds of Theorems 3 and 4.

Lemma 10 *If $\mathcal{Y}$ is the d-dimensional probability simplex and*

$$W(y) = \ln d + \sum_i y_i \ln y_i + I_{\mathcal{Y}}(y)$$

then Equation (13) holds using the norm $\|\cdot\|_1$ and $C = 1/2$. And, Equation (14) holds with $A = \ln d$, $B = 1$, $p = 1$, and $q = \infty$.

Proof: We will verify Equation (13) first. Write

$$W_0(y) = \ln d + \sum_i y_i \ln y_i$$

We have $W_0 = W$ inside $\mathcal{Y}$, and since W_0 is differentiable in all of $\mathbb{R}^d_{++}$ it will be easier to work with. Pick a hypothesis $y \in \text{rel int}\,\mathcal{Y} = \text{dom}\,\partial W$ and a direction Δ with $\|\Delta\|_1 = 1$. Define

$$W_{y,\Delta}(\lambda) = W_0(y + \lambda\Delta)$$

Assume without loss of generality that $\sum_i \Delta_i = 0$ (since Equation (13) holds trivially for $y + \lambda\Delta$ if $\sum_i \Delta_i \neq 0$). Now, Equation (13) with $C = 1/2$, evaluated at hypothesis y and increment $\lambda\Delta$, becomes

$$W(y + \lambda\Delta) \geq W(y) + \lambda\Delta \cdot s + \lambda^2/2 \qquad \forall s \in \partial W(y)$$

Since $\sum_i \Delta_i = 0$, we may without loss of generality take $s = W_0'(y)$. That means that, since $W_{y,\Delta}'(\lambda) = \Delta \cdot W_0'(y + \lambda\Delta)$, we need to show

$$W_{y,\Delta}(\lambda) \geq W_{y,\Delta}(0) + \lambda W_{y,\Delta}'(0) + \lambda^2/2 \tag{39}$$

Equation (39) holds if $W_{y,\Delta}''(\lambda) \geq 1$ for all λ such that $y + \lambda\Delta \in \text{rel int}\,\mathcal{Y}$. To check this condition, we can calculate derivatives of $W_{y,\Delta}$ with respect to λ. The first derivative is

$$\frac{d}{d\lambda} W_{y,\Delta}(\lambda) = \sum_i \Delta_i (1 + \ln(y_i + \lambda\Delta_i))$$

The second derivative is

$$\frac{d^2}{d\lambda^2} W_{y,\Delta}(\lambda) = \sum_i \Delta_i^2/(y_i + \lambda\Delta_i)$$

or, writing $x = y + \lambda\Delta$,

$$\frac{d^2}{d\lambda^2} W_{y,\Delta}(\lambda) = \sum_i \Delta_i^2/x_i \tag{40}$$

We want to verify that the second derivative is always at least 1, so we will find the $x \in \mathbb{R}^d_+$ which makes (40) as small as possible. Since (40) is a convex function of x which approaches ∞ as any component of x approaches 0,[10] the second derivative is smallest

[10]Unless $\Delta_i = 0$ for some i, in which case we can fix $x_i = 0$ and apply the rest of our argument to the remaining components of x. To see why, consider any j such that $\Delta_j^2 > 0$. To reduce (40) we want to make x_j as large as possible. If x_i were positive, we could increase x_j by reducing x_i; so, x_i cannot be positive at the minimum.

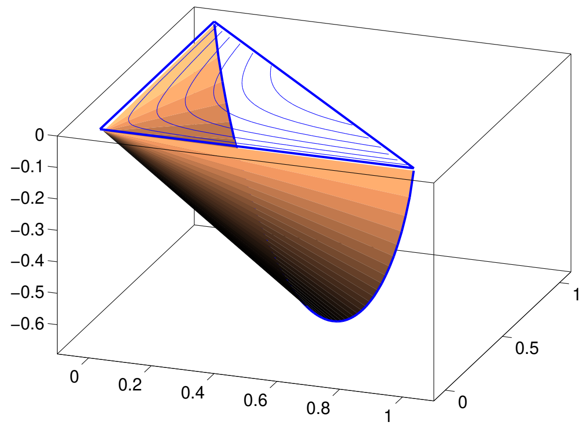

Figure 9: The function $\bar{W}$. $W - \ln d$ is the negentropy function, shown as the solid curve extending downward from $(0, 1, 0)$ and $(1, 0, 0)$, while $\bar{W}$ is the shaded surface. A contour plot of $\bar{W}$ is shown projected on the xy plane. $\bar{W}$ is the greatest convex function which satisfies the conditions (a) $\bar{W}(0) = 0$ and (b) whenever $W(y)$ is finite, $\bar{W}(y) = W(y) - \ln d$.

when the gradient of (40) with respect to x is orthogonal to the constraint $\sum_i x_i = 1$. This happens when there is some constant $k > 0$ such that

$$\Delta_i^2 / x_i^2 = k \qquad \forall i$$

or equivalently $x_i = \sqrt{k}\,|\Delta_i|$. Since $\sum_i |\Delta_i| = 1$ and $\sum_i x_i = 1$, we have $k = 1$ and $x_i = |\Delta_i|$. Substituting back into (40), that means

$$\frac{d^2}{d\lambda^2} W_{y,\Delta}(\lambda) \geq \sum_i \Delta_i^2 / |\Delta_i| = \sum_i |\Delta_i| = 1$$

for all λ. Since y and Δ were arbitrary, we have now verified that W satisfies (13).

For the second part: when $\mathcal{Y}$ is the probability simplex, $\bar{\mathcal{Y}}$ is the positive orthant. Outside the positive orthant, (14) holds trivially. Within the positive orthant, the left-hand side of (14) is

$$\text{conv} \min(W - \ln d + I_{\bar{\mathcal{Y}}},\, I_0) \equiv \bar{W}$$

which is plotted in Figure 9. $\bar{W}$ is negative when $\sum_i y_i \leq 1$, while the right-hand side of (14) is $I_{[-1,1]}(\|y\|_1)$, which is zero when $\sum_i y_i \leq 1$. Both the left-hand and right-hand sides are infinite when $\sum_i y_i > 1$. $\square$

Function	Dual
$I_{[-1,1]}(x)$	$\|y\|$
$I_{[0,1]}(x)$	$[y]_+$
$\|x\|^p/p$	$\|y\|^q/q\ \ (\frac{1}{p}+\frac{1}{q}=1,\ p,q\geq 1)$
$x^p/p + I_{[0,\infty)}(x)$	$[y]_+^q/q\ \ (\frac{1}{p}+\frac{1}{q}=1,\ p,q\geq 1)$
$\sqrt{1+x^2}$	$-\sqrt{1-y^2}\ \ (-1\leq y\leq 1)$
$-\ln x\ \ (x>0)$	$1-\ln(y)\ \ (y>0)$
e^x	$y\ln y - y\ \ (y\geq 0)$
$aF(x)$	$aF^*(y/a)\ \ (a\neq 0)$
$F(ax)$	$F^*(y/a)\ \ (a\neq 0)$
$F(x)+k$	$F^*(y)-k$
$F(x+k)$	$F^*(y)-ky$
$F(x)+G(x)$	$(F^*\ \square\ G^*)(y)$
$\max(F(x),\, G(x))$	$\text{conv}\ \min(F^*(y),\, G^*(y))$

Figure 10: Convex functions and their duals (adapted from [20, 21]).

E Convex duality

This appendix provides some standard notation and results from convex duality which are used in the rest of the paper. For more information on convex duality, Rockafellar's textbook [20] is a good resource; an introduction with a focus on optimization is in Chapters 2–5 of Boyd and Vandenberghe's textbook [19].

A set of points is called *convex* if it contains all weighted averages of its elements, and it is called *closed* if it contains all limits of sequences of its elements. Given a function $F(x)$, define the set

$$\text{epi}(F) = \{(x,z) \mid z \geq F(x)\}$$

which contains the graph of F and the area above that graph. The set epi(F) is called the *epigraph* of F. We will say that the function F is convex iff epi(F) is convex, and closed iff epi(F) is closed. $F(x)$ is allowed to be infinite, in which case epi(F) has no elements of the form (x,z) for any z. The set $\{x \mid F(x) < \infty\}$ is called the domain of F, $\text{dom}\,F$.

Given a function $F(x)$, its *convex dual* is defined as

$$F^*(y) = \sup_x\,(x\cdot y - F(x)) \tag{41}$$

F^* is guaranteed to be closed and convex, and if F is closed and convex then $F^{**} = F$. Any y_0 which satisfies

$$F(x) \geq F(x_0) + (x-x_0)\cdot y_0 \qquad \forall x \tag{42}$$

is called a *subgradient* of F at x_0, written $y_0 \in \partial F(x_0)$. The subgradient exists almost everywhere that F is defined: for example, if x_0 is in the relative interior of $\operatorname{dom} F$, then $\partial F(x_0)$ is nonempty. The subgradients of a differentiable convex function are just its gradients. For any closed convex function F, $x_0 \in \partial F(y_0)$ iff $y_0 \in \partial F^*(x_0)$; that is, the subgradients of F and F^* are inverses of one another.

Convex duality is related to geometric duality: if $F(x) = I_C(x)$ is the indicator function of a cone C, then the dual of F is $F^*(y) = I_{C^\perp}(y)$, where $C^\perp$ is the dual or polar cone to C. The indicator function I_C of a set C is defined by $I_C(x) = 0$ for $x \in C$ and $I_C(x) = \infty$ for $x \notin C$.

Convex duality is also related to duality of seminorms. Let $\|\cdot\|$ and $\|\cdot\|_\circ$ be dual seminorms. Let $\phi : \mathbb{R} \mapsto \mathbb{R}$ be a convex function with $\phi(x) = \phi(-x)$, and suppose ϕ is monotone nondecreasing on $[0, \infty)$. Then the two functions

$$\phi(\|x\|) \qquad\qquad \phi^*(\|y\|_\circ)$$

are dual to each other. (For a proof of the above result, see [20], particularly p. 110 and Theorem 15.3.) As an example, the norms $\|x\|_1 = \sum_i |x_i|$ and $\|x\|_\infty = \max_i |x_i|$ are dual to each other, and we could take $\phi(x) = x^2/2$. In this case, we would have that $\|x\|_1^2/2$ and $\|y\|_\infty^2/2$ are duals.

Figure 10 lists some examples of functions and their duals, including some algebraic rules for computing duals. In the figure, the notation $F \,\square\, G$ means the *infimal convolution* of F and G,

$$(F \,\square\, G)(y) = \inf_z \left(F(y - z) + G(z)\right)$$

Infimal convolution is interesting because it is the dual of addition:

$$(F + G)^* = F^* \,\square\, G^*$$

As an example, if we take $F(x) = \|x\|_1^2$ and $G(x) = I_C(x)$ for a cone C, then $(F + G)^*(y)$ is the squared distance of y from $C^\perp$ using the norm $\|\cdot\|_\infty$.

A final useful fact is that the convex duality operator reverses inequalities between functions: for example, if $F(x) \geq G(x)$ for all x, then $F^*(y) \leq G^*(y)$ for all y.