Machine Learning, 43, 265–291, 2001

Drifting Games

ROBERT E. SCHAPIRE schapire@research.att.com
AT&T Labs—Research, Shannon Laboratory, 180 Park Avenue, Room A279, Florham Park, NJ 07932-0971, USA

Editor: Yoram Singer

Abstract. We introduce and study a general, abstract game played between two players called the shepherd and the adversary. The game is played in a series of rounds using a finite set of "chips" which are moved about in $\mathbb{R}^n$. On each round, the shepherd assigns a desired direction of movement and an importance weight to each of the chips. The adversary then moves the chips in any way that need only be weakly correlated with the desired directions assigned by the shepherd. The shepherd's goal is to cause the chips to be moved to low-loss positions, where the loss of each chip at its final position is measured by a given loss function.

We present a shepherd algorithm for this game and prove an upper bound on its performance. We also prove a lower bound showing that the algorithm is essentially optimal for a large number of chips. We discuss computational methods for efficiently implementing our algorithm.

We show that our general drifting-game algorithm subsumes some well studied boosting and on-line learning algorithms whose analyses follow as easy corollaries of our general result.

Keywords: boosting, on-line learning algorithms

1. Introduction

We introduce a general, abstract game played between two players called the *shepherd*[1] and the *adversary*. The game is played in a series of rounds using a finite set of "chips" which are moved about in $\mathbb{R}^n$. On each round, the shepherd assigns a desired direction of movement to each of the chips, as well as a nonnegative weight measuring the relative importance that each chip be moved in the desired direction. In response, the adversary moves each chip however it wishes, so long as the relative movements of the chips projected in the directions chosen by the shepherd are at least δ, on average. Here, the average is taken with respect to the importance weights that were selected by the shepherd, and $\delta \geq 0$ is a given parameter of the game. Since we think of δ as a small number, the adversary need move the chips in a fashion that is only weakly correlated with the directions desired by the shepherd. The adversary is also restricted to choose relative movements for the chips from a given set $B \subseteq \mathbb{R}^n$. The goal of the shepherd is to force the chips to be moved to low-loss positions, where the loss of each chip at its final position is measured by a given loss function L. A more formal description of the game is given in Section 2.

We present in Section 4 a new algorithm called "OS" for playing this game in the role of the shepherd, and we analyze the algorithm's performance for any parameterization of the game meeting certain natural conditions. Under the same conditions, we also prove in Section 5 that our algorithm is the best possible when the number of chips becomes large.

As spelled out in Section 3, the drifting game is closely related to boosting, the problem of finding a highly accurate classification rule by combining many weak classifiers or hypotheses. The drifting game and its analysis are generalizations of Freund's (1995) "majority-vote game" which was used to derive his boost-by-majority algorithm. This latter algorithm is optimal in a certain sense for boosting binary problems using weak hypotheses which are restricted to making binary predictions. However, the boost-by-majority algorithm has never been generalized to multiclass problems, nor to a setting in which weak hypotheses may "abstain" or give graded predictions between two classes. The general drifting game that we study leads immediately to new boosting algorithms for these settings. By our result on the optimality of the OS algorithm, these new boosting algorithms are also best possible, assuming as we do in this paper that the final hypothesis is restricted in form to a simple majority vote. We do not know if the derived algorithms are optimal without this restriction.

In Section 6, we discuss computational methods for implementing the OS algorithm. We give a useful theorem for handling games in which the loss function enjoys certain monotonicity properties. We also give a more general technique using linear programming for implementing OS in many settings, including the drifting game that corresponds to multiclass boosting. In this latter case, the algorithm runs in polynomial time when the number of classes is held constant.

In Section 7, we discuss the analysis of several drifting games corresponding to previously studied learning problems. For the drifting games corresponding to binary boosting with or without abstaining weak hypotheses, we show how to implement the algorithm efficiently. We also show that there are parameterizations of the drifting game under which OS is equivalent to a simplified version of the AdaBoost algorithm (Freund & Schapire, 1997; Schapire & Singer, 1999), as well as Cesa-Bianchi et al.'s (1996) BW algorithm and Littlestone and Warmuth's (1994) weighted majority algorithm for combining the advice of experts in an on-line learning setting. Analyses of these algorithms follow as easy corollaries of the analysis we give for general drifting games.

2. Drifting games

We begin with a formal description of the drifting game. An outline of the game is shown in figure 1. There are two players in the game called the *shepherd* and the *adversary*. The game is played in T rounds using m *chips*. On each round, the shepherd specifies a *weight vector* $\mathbf{w}_i^t \in \mathbb{R}^n$ for each chip i. The direction of this vector, $\mathbf{v}_i^t = \mathbf{w}_i^t / \|\mathbf{w}_i^t\|_p$, specifies a desired direction of drift, while the length of the vector $\|\mathbf{w}_i^t\|_p$ specifies the relative importance of moving the chip in the desired direction. In response, the adversary chooses a *drift vector* $\mathbf{z}_i^t$ for each chip i. The adversary is constrained to choose each $\mathbf{z}_i^t$ from a fixed set $B \subseteq \mathbb{R}^n$. Moreover, the $\mathbf{z}_i^t$'s must satisfy

$$\sum_i \mathbf{w}_i^t \cdot \mathbf{z}_i^t \geq \delta \sum_i \left\| \mathbf{w}_i^t \right\|_p \tag{1}$$

parameters: number of rounds T
dimension of space n
set $B \subseteq \mathbb{R}^n$ of permitted relative movements
norm l_p where $p \geq 1$
minimum average drift $\delta \geq 0$
loss function $L : \mathbb{R}^n \to \mathbb{R}$
number of chips m

for $t = 1, \ldots, T$:

- shepherd chooses weight vector $\mathbf{w}_i^t \in \mathbb{R}^n$ for each chip i
- adversary chooses drift vector $\mathbf{z}_i^t \in B$ for each chip i so that

$$\sum_{i=1}^{m} \mathbf{w}_i^t \cdot \mathbf{z}_i^t \geq \delta \sum_{i=1}^{m} \left\| \mathbf{w}_i^t \right\|_p$$

the final loss suffered by the shepherd is $\frac{1}{m} \sum_{i=1}^{m} L\left(\sum_{t=1}^{T} \mathbf{z}_i^t \right)$

Figure 1. The drifting game.

or equivalently

$$\frac{\sum_i \left\| \mathbf{w}_i^t \right\|_p \mathbf{v}_i^t \cdot \mathbf{z}_i^t}{\sum_i \left\| \mathbf{w}_i^t \right\|_p} \geq \delta \tag{2}$$

where $\delta \geq 0$ is a fixed parameter of the game. (Here and throughout the paper, when clear from context, $\sum_i$ denotes $\sum_{i=1}^{m}$; likewise, we will shortly use the notation $\sum_t$ for $\sum_{t=1}^{T}$.) In words, $\mathbf{v}_i^t \cdot \mathbf{z}_i^t$ is the amount by which chip i has moved in the desired direction. Thus, the left hand side of Eq. (2) represents a weighted average of the drifts of the chips projected in the desired directions where chip i's projected drift is weighted by $\|\mathbf{w}_i^t\|_p / \sum_i \|\mathbf{w}_i^t\|_p$. We require that this average projected drift be at least δ.

The *position* of chip i at time t, denoted by $\mathbf{s}_i^t$, is simply the sum of the drifts of that chip up to that point in time. Thus, $\mathbf{s}_i^1 = \mathbf{0}$ and $\mathbf{s}_i^{t+1} = \mathbf{s}_i^t + \mathbf{z}_i^t$. The final position of chip i at the end of the game is $\mathbf{s}_i^{T+1}$.

At the end of T rounds, we measure the shepherd's performance using a function L of the final positions of the chips; this function is called the *loss function*. Specifically, the shepherd's goal is to minimize

$$\frac{1}{m} \sum_i L\left(\mathbf{s}_i^{T+1}\right).$$

Summarizing, we see that a game is specified by several parameters: the number of rounds T; the dimension n of the space; a norm $\|\cdot\|_p$ on $\mathbb{R}^n$; a set $B \subseteq \mathbb{R}^n$; a minimum drift constant $\delta \geq 0$; a loss function L; and the number of chips m.

Since the length of weight vectors $\mathbf{w}$ are measured using an l_p-norm, it is natural to measure drift vectors $\mathbf{z}$ using a dual l_q-norm where $1/p + 1/q = 1$. When clear from context, we will generally drop p and q subscripts and write simply $\|\mathbf{w}\|$ or $\|\mathbf{z}\|$.

As an example of a drifting game, suppose that the game is played on the real line and that the shepherd's goal is to get as many chips as possible into the interval $[2, 7]$. Suppose further that the adversary is constrained to move each chip left or right by one unit, and that, on each round, 10% of the chips (as weighted by the shepherd's chosen distribution over chips) must be moved in the shepherd's desired direction. Then for this game, $n = 1$, $B = \{-1, +1\}$ and $\delta = 0.1$. Any norm will do (since we are working in just one dimension), and the loss function is

$$L(s) = \begin{cases} 0 & \text{if } s \in [2, 7] \\ 1 & \text{otherwise.} \end{cases}$$

We will return to this example later in the paper.

Drifting games bear a certain resemblence to the kind of games studied in Blackwell's (1956) celebrated approachability theory. However, it is unclear what the exact relationship is between these two types of games and whether one type is a special case of the other.

3. Relation to boosting

In this section, we describe how the general game of drift relates directly to boosting. In the simplest boosting model, there is a boosting algorithm that has access to a weak learning algorithm that it calls in a series of rounds. There are m given labeled examples $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$ and $y_i \in \{-1, +1\}$. On each round t, the booster chooses a distribution $D_t(i)$ over the examples. The weak learner then must generate a weak hypothesis $h_t : X \to \{-1, +1\}$ whose error is at most $1/2 - \gamma$ with respect to distribution D_t. That is,

$$\Pr_{i \sim D_t}[y_i \neq h_t(x_i)] \leq \tfrac{1}{2} - \gamma. \tag{3}$$

Here, $\gamma > 0$ is known a priori to both the booster and the weak learner. After T rounds, the booster outputs a final hypothesis which we here assume is a majority vote of the weak hypotheses:

$$H(x) = \operatorname{sign}\left(\sum_t h_t(x) \right). \tag{4}$$

For our purposes, the goal of the booster is to minimize the fraction of errors of the final hypothesis on the given set of examples:[2]

$$\frac{1}{m} |\{i : y_i \neq H(x_i)\}|. \tag{5}$$

We can recast boosting as just described as a special-case drifting game; a similar game, called the "majority-vote game," was studied by Freund (1995) for this case. The chips are identified with examples, and the game is one-dimensional so that $n = 1$. The drift of a chip z_i^t is $+1$ if example i is correctly classified by h_t and -1 otherwise; that is, $z_i^t = y_i h_t(x_i)$

and $B = \{-1, +1\}$. The weight w_i^t is formally permitted to be negative, something that does not make sense in the boosting setting; however, for the optimal shepherd described in the next section, this weight will always be nonnegative for this game (by Theorem 7), so we henceforth assume that $w_i^t \geq 0$. The distribution $D_t(i)$ corresponds to $w_i^t / \sum_i w_i^t$. Then the condition in Eq. (3) is equivalent to

$$\sum_i \left[\frac{w_i^t}{\sum_i w_i^t} \left(\frac{1 - z_i^t}{2} \right) \right] \leq \frac{1}{2} - \gamma$$

or

$$\sum_i w_i^t z_i^t \geq 2\gamma \sum_i w_i^t. \tag{6}$$

This is the same as Eq. (1) if we let $\delta = 2\gamma$. Finally, if we define the loss function to be

$$L(s) = \begin{cases} 1 & \text{if } s \leq 0 \\ 0 & \text{if } s > 0 \end{cases} \tag{7}$$

then

$$\frac{1}{m} \sum_i L\left(s_i^{T+1}\right) \tag{8}$$

is exactly equal to Eq. (5).

Our main result on playing drifting games yields in this case exactly Freund's boost-by-majority algorithm (1995). There are numerous variants of this basic boosting setting to which Freund's algorithm has never been generalized and analyzed. For instance, we have so far required weak hypotheses to output values in $\{-1, +1\}$. It is natural to generalize this model to allow weak hypotheses to take values in $\{-1, 0, +1\}$ so that the weak hypotheses may "abstain" on some examples, or to take values in $[-1, +1]$ so that a whole range of values is possible. These correspond to simple modifications of the drifting game described above in which we simply change B to $\{-1, 0, +1\}$ or $[-1, +1]$. As before, we require that Eq. (6) hold for all weak hypotheses and we attempt to design a boosting algorithm which minimizes Eq. (8). For both of these cases, we are able to derive analogs of the boost-by-majority algorithm which we prove are optimal in a particular sense.

Another direction for generalization is to the non-binary multiclass case in which labels y_i belong to a set $Y = \{1, \ldots, n\}$, $n > 2$. Following generalizations of the boosting algorithm AdaBoost to the multiclass case (Freund & Schapire, 1997; Schapire & Singer, 1999), we allow the booster to assign weights both to examples and labels. That is, on each round, the booster devises a distribution $D_t(i, \ell)$ over examples i and labels $\ell \in Y$. The weak learner then computes a weak hypothesis $h_t : X \times Y \to \{-1, +1\}$ which must be correct on a non-trivial fraction of the example-label pairs. That is, if we define

$$\chi_y(\ell) = \begin{cases} +1 & \text{if } y = \ell \\ -1 & \text{otherwise} \end{cases}$$

then we require

$$\Pr_{(i,\ell)\sim D_t}[h_t(x_i, \ell) \neq \chi_{y_i}(\ell)] \leq \tfrac{1}{2} - \gamma. \tag{9}$$

The final hypothesis, we assume, is again a plurality vote of the weak hypotheses:

$$H(x) = \arg\max_{y\in Y} \sum_t h_t(x, y). \tag{10}$$

We can cast this multiclass boosting problem as a drifting game as follows. We have n dimensions, one per class. It will be convenient for the first dimension always to correspond to the correct label, with the remaining $n-1$ dimensions corresponding to incorrect labels. To do this, let us define a map $\pi_\ell : \mathbb{R}^n \to \mathbb{R}^n$ which simply swaps coordinates 1 and ℓ, leaving the other coordinates untouched. The weight vectors $\mathbf{w}_i^t$ correspond to the distribution D_t, modulo swapping of coordinates, a correction of sign and normalization:

$$D_t(i, \ell) = \frac{\left|\left[\pi_{y_i}\left(\mathbf{w}_i^t\right)\right]_\ell\right|}{\sum_i \left\|\mathbf{w}_i^t\right\|}$$

The norm used here to measure weight vectors is l_1-norm. Also, it will follow from Theorem 7 that, for optimal play of this game, the first coordinate of $\mathbf{w}_i^t$ is always nonnegative and all other coordinates are nonpositive. The drift vectors $\mathbf{z}_i^t$ are derived as before from the weak hypotheses:

$$\mathbf{z}_i^t = \pi_{y_i}(\langle h_t(x_i, 1), \ldots, h_t(x_i, n)\rangle).$$

It can be verified that the condition in Eq. (9) is equivalent to Eq. (1) with $\delta = 2\gamma$. For binary weak hypotheses, $B = \{-1, +1\}^n$.

The final hypothesis H makes a mistake on example (x, y) if and only if

$$\sum_t h_t(x, y) \leq \max_{\ell:\ell\neq y} \sum_t h_t(x, \ell).$$

Therefore, we can count the fraction of mistakes of the final hypothesis in the drifting game context as

$$\frac{1}{m}\sum_i L\left(\mathbf{s}_i^{T+1}\right)$$

where

$$L(\mathbf{s}) = \begin{cases} 1 & \text{if } s_1 \leq \max\{s_2, \ldots, s_n\} \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

Thus, by giving an algorithm for the general drifting game, we also obtain a generalization of the boost-by-majority algorithm for multiclass problems. The algorithm can be implemented in this case in polynomial time for a constant number of classes n, and the algorithm is provably best possible in a particular sense.

We note also that a simplified form of the AdaBoost algorithm (Freund & Schapire, 1997; Schapire & Singer, 1999) can be derived as an instance of the OS algorithm simply by changing the loss function L in Eq. (7) to an exponential $L(s) = \exp(-\eta s)$ for some $\eta > 0$. More details on this game are given in Section 7.2.

Besides boosting problems, the drifting game also generalizes the problem of learning on-line with a set of "experts" (Cesa-Bianchi et al., 1997; Littlestone & Warmuth, 1994). In particular, the BW algorithm of Cesa-Bianchi et al. (1996) and the weighted majority algorithm of Littestone and Warmuth (1994) can be derived as special cases of our main algorithm for a particular natural parameterization of the drifting game. Details are given in Section 7.3.

4. The algorithm and its analysis

We next describe our algorithm for playing the general drifting game of Section 2. Like Freund's boost-by-majority algorithm (1995), the algorithm we present here uses a "potential function" which is central both to the workings of the algorithm and its analysis. This function can be thought of as a "guess" of the loss that we expect to suffer for a chip at a particular position and at a particular point in time.

We denote the potential of a chip at position $\mathbf{s}$ on round t by $\phi_t(\mathbf{s})$. The final potential is the actual loss so that $\phi_T = L$. The potential functions ϕ_t for earlier time steps are defined inductively:

$$\phi_{t-1}(\mathbf{s}) = \min_{\mathbf{w}\in\mathbb{R}^n} \sup_{\mathbf{z}\in B} (\phi_t(\mathbf{s}+\mathbf{z}) + \mathbf{w}\cdot\mathbf{z} - \delta\|\mathbf{w}\|_p). \tag{12}$$

We will show later that, under natural conditions, the minimum above actually exists. Moreover, the minimizing vector $\mathbf{w}$ is the one used by the shepherd for the algorithm we now present. We call our shepherd algorithm "OS" for "optimal shepherd." The weight vector $\mathbf{w}_i^t$ chosen by OS for chip i is any vector $\mathbf{w}$ which minimizes

$$\sup_{\mathbf{z}\in B} \left(\phi_t\left(\mathbf{s}_i^t + \mathbf{z}\right) + \mathbf{w}\cdot\mathbf{z} - \delta\|\mathbf{w}\|_p\right).$$

Returning to the example at the end of Section 2, figure 2 shows the potential function ϕ_t and the weights that would be selected by OS as a function of the position of each chip for various choices of t. For this figure, $T = 20$.

We will need some natural assumptions to analyze this algorithm. The first assumption states merely that the allowed drift vectors in B are bounded; for convenience, we assume they have norm at most one.

Assumption 1. $\sup_{\mathbf{z}\in B} \|\mathbf{z}\|_q \le 1$.

We next assume that the loss function L is bounded.

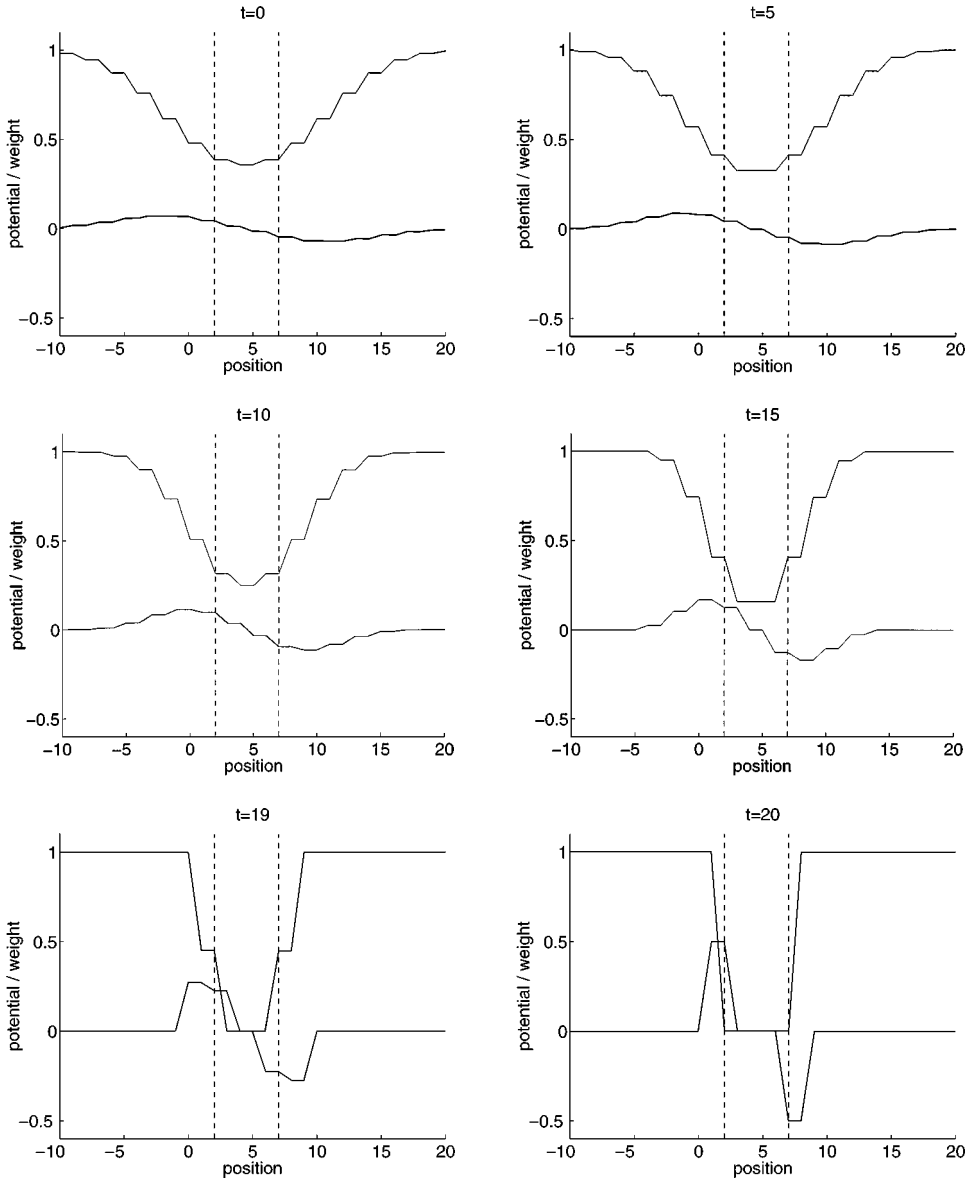

Figure 2. Plots of the potential function (top curve in each figure) and the weights selected by OS (bottom curves) as a function of the position of a chip in the example game at the end of Section 2 for various choices of t and with $T = 20$. The vertical dotted lines show the boundary of the goal interval $[2, 7]$. Curves are only meaningful at integer values.

Assumption 2. There exist finite $L_{\min}$ and $L_{\max}$ such that $L_{\min} \leq L(\mathbf{s}) \leq L_{\max}$ for all $\mathbf{s} \in \mathbb{R}^n$.

In fact, this assumption need only hold for all $\mathbf{s}$ with $\|\mathbf{s}\|_q \leq T$ since positions outside this range are never reached, given Assumption 1.

Finally, we assume that, for any direction $\mathbf{v}$, it is possible to choose a drift whose projection onto $\mathbf{v}$ is more than δ by a constant amount.

Assumption 3. There exists a number $\mu > 0$ such that for all $\mathbf{w} \in \mathbb{R}^n$ there exists $\mathbf{z} \in B$ with $\mathbf{w} \cdot \mathbf{z} \geq (\delta + \mu)\|\mathbf{w}\|$.

Lemma 1. *Given Assumptions* 1, 2 *and* 3, *for all* $t = 0, \ldots, T$:
1. *the minimum in Eq.* (12) *exists*; *and*
2. $L_{\min} \leq \phi_t(\mathbf{s}) \leq L_{\max}$ *for all* $\mathbf{s} \in \mathbb{R}^n$.

Proof: By backwards induction on t. The base cases are trivial. Let us fix $\mathbf{s}$ and let $F(\mathbf{z}) = \phi_t(\mathbf{s} + \mathbf{z})$. Let

$$H(\mathbf{w}) = \sup_{\mathbf{z} \in B} (F(\mathbf{z}) + \mathbf{w} \cdot \mathbf{z} - \delta\|\mathbf{w}\|).$$

Using Assumption 1, for any $\mathbf{w}$, $\mathbf{w}'$:

$$\begin{aligned} |H(\mathbf{w}') - H(\mathbf{w})| &\leq \sup_{\mathbf{z} \in B} |(F(\mathbf{z}) + \mathbf{w} \cdot \mathbf{z} - \delta\|\mathbf{w}\|) - (F(\mathbf{z}) + \mathbf{w}' \cdot \mathbf{z} - \delta\|\mathbf{w}'\|)| \\ &= \sup_{\mathbf{z} \in B} |(\mathbf{w} - \mathbf{w}') \cdot \mathbf{z} + \delta(\|\mathbf{w}'\| - \|\mathbf{w}\|)| \\ &\leq (1 + \delta)\|\mathbf{w}' - \mathbf{w}\|. \end{aligned}$$

Therefore, H is continuous. Moreover, for $\mathbf{w} \in \mathbb{R}^n$, by Assumptions 2 and 3 (as well as our inductive hypothesis),

$$H(\mathbf{w}) \geq L_{\min} + (\delta + \mu)\|\mathbf{w}\| - \delta\|\mathbf{w}\| = L_{\min} + \mu\|\mathbf{w}\|. \tag{13}$$

Since

$$H(\mathbf{0}) \leq L_{\max}, \tag{14}$$

it follows that $H(\mathbf{w}) > H(\mathbf{0})$ if $\|\mathbf{w}\| > (L_{\max} - L_{\min})/\mu$. Thus, for computing the minimum of H,we only need consider points in the compact set

$$\left\{ \mathbf{w} : \|\mathbf{w}\| \leq \frac{L_{\max} - L_{\min}}{\mu} \right\}.$$

Since a continuous function over a compact set has a minimum, this proves Part 1.

Part 2 follows immediately from Eqs. (13) and (14). □

We next prove an upper bound on the loss suffered by a shepherd employing the OS algorithm against any adversary. This is the main result of this section. We will shortly see that this bound is essentially best possible for any algorithm. It is important to note that these theorems tell us much more than the almost obvious point that the optimal thing to do is whatever is best in a minmax sense. These theorems prove the nontrivial fact that (nearly) minmax behavior can be obtained without the simultaneous consideration of all of the chips at once. Rather, we can compute each weight vector $\mathbf{w}_i^t$ merely as a function of the position of chip i, without consideration of the positions of any of the other chips.

Theorem 2. *Under the condition of Assumptions* 1–3, *the final loss suffered by the OS algorithm against any adversary is at most* $\phi_0(\mathbf{0})$ *where the functions* ϕ_t *are defined above.*

Proof: Following Freund's analysis (1995), we show that the total potential never increases. That is, we prove by induction that

$$\sum_i \phi_t\left(\mathbf{s}_i^{t+1}\right) \le \sum_i \phi_{t-1}\left(\mathbf{s}_i^t\right). \tag{15}$$

This implies, through repeated application of Eq. (15), that

$$\frac{1}{m}\sum_i L\left(\mathbf{s}_i^{T+1}\right) = \frac{1}{m}\sum_i \phi_T\left(\mathbf{s}_i^{T+1}\right) \le \frac{1}{m}\sum_i \phi_0\left(\mathbf{s}_i^1\right) = \phi_0(\mathbf{0})$$

as claimed.

The definition of ϕ_{t-1} given in Eq. (12) implies that for $\mathbf{w}_i^t$ chosen by the OS algorithm, and for all $\mathbf{z} \in B$ and all $\mathbf{s} \in \mathbb{R}^n$:

$$\phi_t(\mathbf{s}+\mathbf{z}) + \mathbf{w}_i^t \cdot \mathbf{z} - \delta\left\|\mathbf{w}_i^t\right\| \le \phi_{t-1}(\mathbf{s}).$$

Therefore,

$$\begin{aligned}\sum_i \phi_t\left(\mathbf{s}_i^{t+1}\right) &= \sum_i \phi_t\left(\mathbf{s}_i^t + \mathbf{z}_i^t\right) \\ &\le \sum_i \left(\phi_{t-1}\left(\mathbf{s}_i^t\right) - \mathbf{w}_i^t \cdot \mathbf{z}_i^t + \delta\left\|\mathbf{w}_i^t\right\|\right) \\ &\le \sum_i \phi_{t-1}\left(\mathbf{s}_i^t\right)\end{aligned}$$

where the last inequality follows from Eq. (1). □

Returning again to the example at the end of Section 2, figure 3 shows a plot of the bound $\phi_0(0)$ as a function of the total number of rounds T. It is rather curious that the bound is not monotonic in T (even discounting the jagged nature of the curve caused by the difference between even and odd length games). Apparently, for this game, having more time to get the chips into the goal region can actually hurt the shepherd.

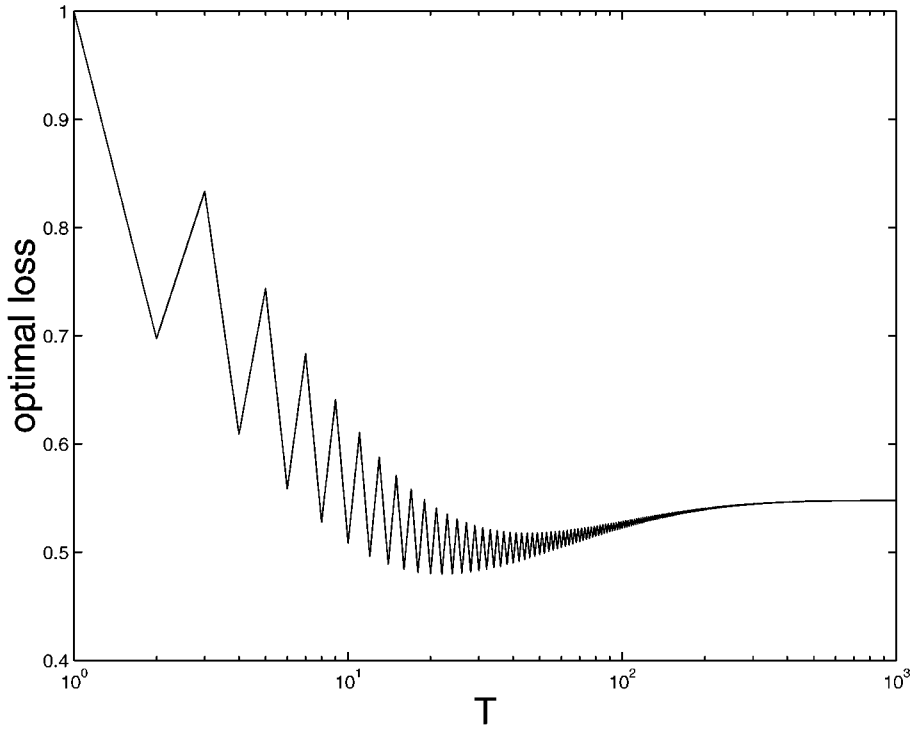

Figure 3. A plot of the loss bound $\phi_0(0)$ as a function of the total number of rounds T for the example game at the end of Section 2. The jagged nature of the curve is due to the difference between a game with an odd or an even number of steps.

5. A lower bound

In this section, we prove that the OS algorithm is essentially optimal in the sense that, for any shepherd algorithm, there exists an adversary capable of forcing a loss matching the upper bound of Theorem 3 in the limit of a large number of chips. Specifically, we prove the following theorem, the main result of this section:

Theorem 2. *Let A be any shepherd algorithm for playing a drifting game satisfying Assumptions* 1–3 *where all parameters of the game are fixed, except the number of chips m. Let ϕ_t be as defined above. Then for any $\epsilon > 0$, there exists an adversary such that for m sufficiently large, the loss suffered by algorithm A is at least $\phi_0(\mathbf{0}) - \epsilon$.*

To prove the theorem, we will need two lemmas. The first gives an abstract result on computing a minimax of the kind appearing in Eq. (12). The second lemma uses the first to prove a characterization of ϕ_t in a form amenable to use in the proof of Theorem 3.

Lemma 4. *Let S be any nonempty, bounded subset of $\mathbb{R}^2$. Let C be the convex hull of S. Then*

$$\inf_{\alpha\in\mathbb{R}} \sup\{y + \alpha x : (x, y) \in S\} = \sup\{y : (0, y) \in C\}.$$

Proof: Let $\bar{C}$ be the closure of C. First, for any $\alpha \in \mathbb{R}$,

$$\begin{aligned}\sup\{y + \alpha x : (x, y) \in S\} &= \sup\{y + \alpha x : (x, y) \in C\} \\ &= \sup\{y + \alpha x : (x, y) \in \bar{C}\}. \end{aligned} \tag{16}$$

The first equality follows from the fact that, if $(x, y) \in C$ then

$$(x, y) = \sum_{i=1}^{N} p_i (x_i, y_i)$$

for some positive integer N, $p_i \in [0, 1]$, $\sum_i p_i = 1$, $(x_i, y_i) \in S$. But then

$$y + \alpha x = \sum_{i=1}^{N} p_i (y_i + \alpha x_i) \le \max_i (y_i + \alpha x_i).$$

The second equality in Eq. (16) follows simply because the supremum of a continuous function on any set is equal to its supremum over the closure of the set. For this same reason,

$$\sup\{y : (0, y) \in C\} = \sup\{y : (0, y) \in \bar{C}\}. \tag{17}$$

Because $\bar{C}$ is closed, convex and bounded, and because the function $y+\alpha x$ is continuous, concave in (x, y) and convex in α, we can reverse the order of the "inf sup" (see, for instance, Corollary 37.3.2 of Rockafellar (1970)). That is,

$$\inf_{\alpha\in\mathbb{R}} \sup_{(x,y)\in\bar{C}} (y + \alpha x) = \sup_{(x,y)\in\bar{C}} \inf_{\alpha\in\mathbb{R}} (y + \alpha x). \tag{18}$$

Clearly, if $x \neq 0$ then

$$\inf_{\alpha\in\mathbb{R}} (y + \alpha x) = -\infty.$$

Thus, the right hand side of Eq. (18) is equal to

$$\sup\{y : (0, y) \in \bar{C}\}.$$

Combining with Eqs. (16) and (17) immediately gives the result. □

Lemma 5. *Under the condition of Assumptions* 1–3, *and for* ϕ_t *as defined above,*

$$\phi_{t-1}(\mathbf{s}) = \inf_{\mathbf{v}:\,\|\mathbf{v}\|=1} \sup \sum_{j=1}^{N} d_j \phi_t(\mathbf{s} + \mathbf{z}_j)$$

where the supremum is taken over all positive integers N, *all* $\mathbf{z}_1, \ldots, \mathbf{z}_N \in B$ *and all non-negative* $d_1, \ldots, d_N$ *satisfying* $\sum_j d_j = 1$ *and*

$$\sum_j d_j \mathbf{v} \cdot \mathbf{z}_j = \delta.$$

Proof: To simplify notation, let us fix t and $\mathbf{s}$. Let F and H be as defined in the proof of Lemma 1. For $\|\mathbf{v}\| = 1$, let

$$G(\mathbf{v}) = \sup \sum_{j=1}^{N} d_j F(\mathbf{z}_j) \tag{19}$$

where again the supremum is taken over d_j's and $\mathbf{z}_j$'s as in the statement of the lemma. Note that by Assumption 3, this supremum cannot be vacuous. Throughout this proof, we use $\mathbf{v}$ to denote a vector of norm one, while $\mathbf{w}$ is a vector of unrestricted norm. Our goal is to show that

$$\inf_{\mathbf{v}} G(\mathbf{v}) = \inf_{\mathbf{w}} H(\mathbf{w}). \tag{20}$$

Let us fix $\mathbf{v}$ momentarily. Let

$$S = \{(\mathbf{v} \cdot \mathbf{z} - \delta,\, F(\mathbf{z})) : \mathbf{z} \in B\}.$$

Then S is bounded by Assumptions 1–3 (and part 2 of Lemma 1), so we can apply Lemma 4 which gives

$$\inf_{\alpha \in \mathbb{R}} \sup_{\mathbf{z} \in B} (F(\mathbf{z}) + \alpha(\mathbf{v} \cdot \mathbf{z} - \delta)) = G(\mathbf{v}). \tag{21}$$

Note that

$$\begin{aligned} \inf_{\alpha \geq 0} H(\alpha \mathbf{v}) &= \inf_{\alpha \geq 0} \sup_{\mathbf{z} \in B} (F(\mathbf{z}) + \alpha \mathbf{v} \cdot \mathbf{z} - \alpha\delta) \\ &\geq \inf_{\alpha \in \mathbb{R}} \sup_{\mathbf{z} \in B} (F(\mathbf{z}) + \alpha \mathbf{v} \cdot \mathbf{z} - \alpha\delta) \\ &\geq \inf_{\alpha \in \mathbb{R}} \sup_{\mathbf{z} \in B} (F(\mathbf{z}) + \alpha \mathbf{v} \cdot \mathbf{z} - |\alpha|\delta) = \inf_{\alpha \in \mathbb{R}} H(\alpha \mathbf{v}) \end{aligned}$$

(where the second inequality uses $\alpha \leq |\alpha|$). Combining with Eq. (21) gives

$$\inf_{\mathbf{v}} \inf_{\alpha \geq 0} H(\alpha \mathbf{v}) \geq \inf_{\mathbf{v}} G(\mathbf{v}) \geq \inf_{\mathbf{v}} \inf_{\alpha \in \mathbb{R}} H(\alpha \mathbf{v}).$$

Since the left and right terms are both equal to $\inf_{\mathbf{w}} H(\mathbf{w})$, this implies Eq. (20) and completes the proof. □

Proof of Theorem 3: We will show that, for m sufficiently large, on round t, the adversary can choose the $\mathbf{z}_i^t$'s so that

$$\frac{1}{m}\sum_i \phi_t\left(\mathbf{s}_i^{t+1}\right) \geq \frac{1}{m}\sum_i \phi_{t-1}\left(\mathbf{s}_i^t\right) - \frac{\epsilon}{T}. \tag{22}$$

Repeatedly applying Eq. (22) implies that

$$\frac{1}{m}\sum_i L\left(\mathbf{s}_i^{T+1}\right) = \frac{1}{m}\sum_i \phi_T\left(\mathbf{s}_i^{T+1}\right) \geq \frac{1}{m}\sum_i \phi_0\left(\mathbf{s}_i^1\right) - \epsilon = \phi_0(\mathbf{0}) - \epsilon$$

proving the theorem.

Fix t. We use a random construction to show that there exist $\mathbf{z}_i^t$'s with the desired properties. For each weight vector $\mathbf{w}_i^t$ chosen by the shepherd, let $d_{i1}, \ldots, d_{iN} \in [0, 1]$ and $\mathbf{z}_{i1}, \ldots, \mathbf{z}_{iN} \in B$ be such that $\sum_j d_{ij} = 1$,

$$\sum_j d_{ij}\mathbf{w}_i^t \cdot \mathbf{z}_{ij} = \delta\left\|\mathbf{w}_i^t\right\|$$

and

$$\sum_j d_{ij}\phi_t\left(\mathbf{s}_i^t + \mathbf{z}_{ij}\right) \geq \phi_{t-1}\left(\mathbf{s}_i^t\right) - \frac{\epsilon}{2T}.$$

Such d_{ij}'s and $\mathbf{z}_{ij}$'s must exist by Lemma 5. Using Assumption 3, let $\mathbf{z}_{i0}$ be such that

$$\mathbf{w}_i^t \cdot \mathbf{z}_{i0} \geq (\delta + \mu)\left\|\mathbf{w}_i^t\right\|.$$

Finally, let $\mathbf{Z}_i$ be a random variable that is $\mathbf{z}_{i0}$ with probability α and $\mathbf{z}_{ij}$ with probability $(1-\alpha)d_{ij}$ (independent of the other $\mathbf{Z}_i$'s). Here,

$$\alpha = \frac{\epsilon}{4T(L_{\max} - L_{\min})}.$$

Let $\mathbf{v}_i = \mathbf{w}_i^t/\|\mathbf{w}_i^t\|$, and let $a_i = \|\mathbf{w}_i^t\|/\sum_i \|\mathbf{w}_i^t\|$. By Assumption 1, $|\mathbf{v}_i \cdot \mathbf{Z}_i| \leq 1$. Also,

$$\mathrm{E}[\mathbf{v}_i \cdot \mathbf{Z}_i] \geq (1-\alpha)\delta + \alpha(\delta + \mu) = \delta + \alpha\mu.$$

Thus, by Hoeffding's inequality (1963),

$$\Pr\left[\sum_i a_i \mathbf{v}_i \cdot \mathbf{Z}_i < \delta\right] \leq \exp\left(-\frac{\alpha^2\mu^2}{2\sum_i a_i^2}\right) \leq e^{-\alpha^2\mu^2/2}. \tag{23}$$

Let $S = (1/m)\sum_i \phi_t(\mathbf{s}_i^t + \mathbf{Z}_i)$. Then

$$\begin{aligned}\mathrm{E}[S] &\geq \frac{1}{m}\sum_i \left[\left(\phi_{t-1}(\mathbf{s}_i^t) - \frac{\epsilon}{2T}\right)(1-\alpha) + \alpha\phi_t(\mathbf{s}_i^t + \mathbf{z}_{i0})\right] \\ &= \frac{1}{m}\sum_i \left[\phi_{t-1}(\mathbf{s}_i^t) + \alpha\left(\phi_t(\mathbf{s}_i^t + \mathbf{z}_{i0}) - \phi_{t-1}(\mathbf{s}_i^t)\right)\right] - \frac{\epsilon}{2T}(1-\alpha) \\ &\geq \frac{1}{m}\sum_i \phi_{t-1}(\mathbf{s}_i^t) - \alpha(L_{\max} - L_{\min}) - \frac{\epsilon}{2T}. \end{aligned} \tag{24}$$

By Hoeffding's inequality (1963), since $L_{\min} \leq \phi_t(\mathbf{s}_i^t + \mathbf{Z}_i) \leq L_{\max}$,

$$\Pr[S < \mathrm{E}[S] - \alpha(L_{\max} - L_{\min})] \leq e^{-2\alpha^2 m}. \tag{25}$$

Now let m be so large that

$$e^{-2\alpha^2 m} + e^{-\alpha^2\mu^2/2} < 1.$$

Then by Eqs. (23) and (25), there exists a choice of $\mathbf{z}_i^t$'s such that

$$\sum_i \mathbf{w}_i^t \cdot \mathbf{z}_i^t = \sum_i a_i \mathbf{v}_i \cdot \mathbf{z}_i^t \geq \delta$$

and such that

$$\begin{aligned}\frac{1}{m}\sum_i \phi_t(\mathbf{s}_i^{t+1}) &= \frac{1}{m}\sum_i \phi_t(\mathbf{s}_i^t + \mathbf{z}_i^t) \\ &\geq \mathrm{E}[S] - \alpha(L_{\max} - L_{\min}) \\ &\geq \frac{1}{m}\sum_i \phi_{t-1}(\mathbf{s}_i^t) - \frac{\epsilon}{T}\end{aligned}$$

by Eq. (24) and our choice of α.

6. Computational methods

In this section, we discuss general computational methods for implementing the OS algorithm.

6.1. Unate loss functions

We first note that, for loss functions L with certain monotonicity properties, the quadrant in which the minimizing weight vectors are to be found can be determined a priori. This often simplifies the search for minima. To be more precise, for $\boldsymbol{\sigma} \in \{-1, +1\}^n$ and $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$,

let us write $\mathbf{x} \geq_\sigma \mathbf{y}$ if $\sigma_i x_i \geq \sigma_i y_i$ for all $1 \leq i \leq n$. We say that a function $f : \mathbb{R}^n \to \mathbb{R}$ is *unate with sign vector* $\boldsymbol{\sigma} \in \{-1, +1\}^n$ if $f(\mathbf{x}) \geq f(\mathbf{y})$ whenever $\mathbf{x} \geq_\sigma \mathbf{y}$.

Lemma 6. *If the loss function L is unate with sign vector $\boldsymbol{\sigma} \in \{-1, +1\}^n$, then so is ϕ_t (as defined above) for $t = 0, \ldots, T$.*

Proof: By backwards induction on t. The base case is immediate. Let $\mathbf{x} \geq_\sigma \mathbf{y}$. Then for any $\mathbf{z} \in B$ and $\mathbf{w} \in \mathbb{R}^n$, $\mathbf{x} + \mathbf{z} \geq_\sigma \mathbf{y} + \mathbf{z}$, and so

$$\phi_t(\mathbf{x} + \mathbf{z}) + \mathbf{w} \cdot \mathbf{z} - \delta\|\mathbf{w}\| \geq \phi_t(\mathbf{y} + \mathbf{z}) + \mathbf{w} \cdot \mathbf{z} - \delta\|\mathbf{w}\|$$

by inductive hypothesis. Therefore, $\phi_{t-1}(\mathbf{x}) \geq \phi_{t-1}(\mathbf{y})$, and so ϕ_{t-1} is also unate. □

For the main theorem of this subsection, we need one more assumption:

Assumption 4. If $\mathbf{z} \in B$ and if $\mathbf{z}'$ is such that $|z'_i| = |z_i|$ for all i, then $\mathbf{z}' \in B$.

Theorem 7. *Under the condition of Assumptions* 1–4, *if L is unate with sign vector $\boldsymbol{\sigma} \in \{-1, +1\}^n$, then for any $\mathbf{s} \in \mathbb{R}^n$, there is a vector $\mathbf{w}$ which minimizes*

$$\sup_{\mathbf{z} \in B} (\phi_t(\mathbf{s} + \mathbf{z}) + \mathbf{w} \cdot \mathbf{z} - \delta\|\mathbf{w}\|)$$

and for which $\mathbf{w} \leq_\sigma \mathbf{0}$.

Proof: Let F and H be as in the proof of Lemma 1. By Lemma 6, F is unate. Let $\mathbf{w} \in \mathbb{R}^n$ have some coordinate i for which $\sigma_i w_i > 0$ so that $\mathbf{w} \not\leq_\sigma \mathbf{0}$. Let $\mathbf{w}'$ be such that

$$w'_j = \begin{cases} w_j & \text{if } j \neq i \\ -w_i & \text{if } j = i. \end{cases}$$

We show that $H(\mathbf{w}') \leq H(\mathbf{w})$. Let $\mathbf{z} \in B$. If $\sigma_i z_i > 0$ then

$$F(\mathbf{z}) + \mathbf{w} \cdot \mathbf{z} - \delta\|\mathbf{w}\| \geq F(\mathbf{z}) + \mathbf{w}' \cdot \mathbf{z} - \delta\|\mathbf{w}'\|.$$

If $\sigma_i z_i \leq 0$ then let $\mathbf{z}'$ be defined analogously to $\mathbf{w}'$. By Assumption 4, $\mathbf{z}' \in B$. Then $\mathbf{z} \leq_\sigma \mathbf{z}'$ and so $F(\mathbf{z}) \leq F(\mathbf{z}')$. Thus,

$$F(\mathbf{z}') + \mathbf{w} \cdot \mathbf{z}' - \delta\|\mathbf{w}\| \geq F(\mathbf{z}) + \mathbf{w}' \cdot \mathbf{z} - \delta\|\mathbf{w}'\|.$$

Hence, $H(\mathbf{w}') \leq H(\mathbf{w})$.

Applying this argument repeatedly, we can derive a vector $\bar{\mathbf{w}}$ with $\bar{\mathbf{w}} \leq_\sigma \mathbf{0}$ and such that $H(\bar{\mathbf{w}}) \leq H(\mathbf{w})$. This proves the theorem. □

Note that the loss functions for all of the games in Section 3 are unate (and also satisfy Assumptions 1–4). The same will be true of all of the games discussed in Section 7. Thus,

for all of these games, we can determine a priori the signs of each of the coordinates of the minimizing vectors used by the OS algorithm.

6.2. A general technique using linear programming

In many cases, we can use linear programming to implement OS. In particular, let us assume that we measure weight vectors $\mathbf{w}$ using the l_1 norm (i.e., $p = 1$). Also, let us assume that $|B|$ is finite. Then given ϕ_t and $\mathbf{s}$, computing

$$\phi_{t-1}(\mathbf{s}) = \min_{\mathbf{w}\in\mathbb{R}^n} \max_{\mathbf{z}\in B} (\phi_t(\mathbf{s}+\mathbf{z}) + \mathbf{w}\cdot\mathbf{z} - \delta\|\mathbf{w}\|)$$

can be rewritten as an optimization problem:

variables: $\mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R}$
minimize: b
subject to: $\forall \mathbf{z} \in B : \phi_t(\mathbf{s}+\mathbf{z}) + \mathbf{w}\cdot\mathbf{z} - \delta\|\mathbf{w}\| \le b.$

The minimizing value b is the desired value of $\phi_{t-1}(\mathbf{s})$. Note that, with respect to the variables $\mathbf{w}$ and b, this problem is "almost" a linear program, if not for the norm operator. However, when L is unate with sign vector $\boldsymbol{\sigma}$, and when the other conditions of Theorem 7 hold, we can restrict $\mathbf{w}$ so that $\mathbf{w} \le_\sigma \mathbf{0}$. This allows us to write

$$\|\mathbf{w}\|_1 = -\sum_{i=1}^{n} \sigma_i w_i.$$

Adding $\mathbf{w} \le_\sigma \mathbf{0}$ as a constraint (or rather, a set of n constraints), we now have derived a linear program with $n+1$ variables and $|B|+n$ constraints. This can be solved in polynomial time.

Thus, for instance, this technique can be applied to the multiclass boosting problem discussed in Section 3. In this case, $B = \{-1, +1\}^n$. So, for any $\mathbf{s}$, $\phi_{t-1}(\mathbf{s})$ can be computed from ϕ_t in time polynomial in 2^n which may be reasonable for small n. In addition, ϕ_t must be computed at each reachable position $\mathbf{s}$ in an n-dimensional integer grid of radius t, i.e., for all $\mathbf{s} \in \{-t, -t+1, \ldots, t-1, t\}^n$. This involves computation of ϕ_t at $(2t+1)^n$ points, giving an overall running time for the algorithm which is polynomial in $(2T+1)^n$. Again, this may be reasonable for very small n. It is an open problem to find a way to implement the algorithm more efficiently.

7. Deriving old and new algorithms

In this section, we show how a number of old and new boosting and on-line learning algorithms can be derived and analyzed as instances of the OS algorithm for appropriately chosen drifting games.

7.1. *Boost-by-majority and variants*

We begin with the drifting game described in Section 3 corresponding to binary boosting with $B = \{-1, +1\}$. For this game,

$$\phi_{t-1}(s) = \min_{w \geq 0} \max\{\phi_t(s-1) - w - \delta w, \phi_t(s+1) + w - \delta w\}$$

where we know from Theorem 7 that only nonnegative values of w need to be considered. It can be argued that the minimum must occur when

$$\phi_t(s-1) - w - \delta w = \phi_t(s+1) + w - \delta w,$$

i.e., when

$$w = \frac{\phi_t(s-1) - \phi_t(s+1)}{2}. \tag{26}$$

This gives

$$\phi_{t-1}(s) = \frac{1+\delta}{2}\phi_t(s+1) + \frac{1-\delta}{2}\phi_t(s-1).$$

Solving gives

$$\phi_t(s) = 2^{t-T} \sum_{0 \leq k \leq (T-t-s)/2} \binom{T-t}{k} \left(\frac{1+\delta}{1-\delta}\right)^k$$

(where we follow the convention that $\binom{n}{k} = 0$ if $k < 0$ or $k > n$). Weighting examples using Eq. (26) gives exactly Freund's (1995) boost-by-majority algorithm (the "boosting by resampling" version).

When $B = \{-1, 0, +1\}$, a similar but more involved analysis gives

$$\phi_{t-1}(s) = \max\left\{(1-\delta)\phi_t(s) + \delta\phi_t(s+1), \frac{1+\delta}{2}\phi_t(s+1) + \frac{1-\delta}{2}\phi_t(s-1)\right\} \tag{27}$$

and the corresponding choice of w is $\phi_t(s) - \phi_t(s+1)$ or $(\phi_t(s-1) - \phi_t(s+1))/2$, depending on whether the maximum in Eq. (27) is realized by the first or second quantity. We do not know how to solve the recurrence in Eq. (27) so that the bound $\phi_0(\mathbf{0})$ given in Theorem 2 can be put in explicit form. Nevertheless, this bound can easily be evaluated numerically, and the algorithm can certainly be implemented efficiently in its present form.

We have thus far been unable to solve the recurrence for the case that $B = [-1, +1]$, even to a point at which the algorithm can be implemented. However, this case can be approximated by the case in which $B = \{i/N : i = -N, \ldots, N\}$ for a moderate value

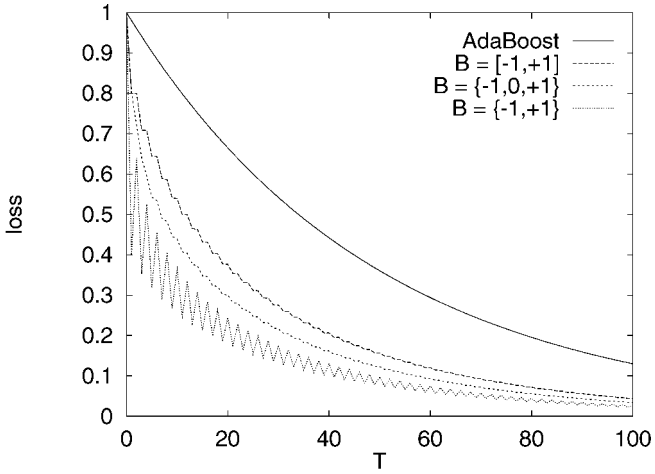

Figure 4. A comparison of the bound $\phi_0(\mathbf{0})$ for the drifting games associated with AdaBoost (Section 7.2) and boost-by-majority (Sections 3 and 7.1). For AdaBoost, η is set as in Eq. (28). For boost-by-majority, the bound is plotted when B is $\{-1, +1\}$, $\{-1, 0, +1\}$ and $[-1, +1]$. (The latter case is approximated by $B = \{i/100 : i = -100, \ldots, 100\}$.) The bound is plotted as a function of the number of rounds T. The drift parameter is fixed to $\delta = 0.2$. (The jagged nature of the $B = \{-1, +1\}$ curve is due to the fact that games with an even number of rounds—in which ties count as a loss for the shepherd so that $L(0) = 1$—are harder than games with an odd number of rounds.)

of N. In the latter case, the potential function and associated weights can be computed numerically. For instance, linear programming can be used as discussed in Section 6.2. Alternatively, it can be shown that Lemma 5 combined with Theorem 7 implies that

$$\phi_{t-1}(s) = \max\{p\phi_t(s + z_1) + (1 - p)\phi_t(s + z_2) : \\ z_1, z_2 \in B,\ p \in [0, 1],\ pz_1 + (1 - p)z_2 = \delta\}$$

which can be evaluated using a simple search over all pairs z_1, z_2 (since B is finite).

Figure 4 compares the bound $\phi_0(\mathbf{0})$ for the drifting games associated with boost-by-majority and variants in which B is $\{-1, +1\}$, $\{-1, 0, +1\}$ and $[-1, +1]$ (using the approximation that was just mentioned), as well as AdaBoost (discussed in the next section). These bounds are plotted as a function of the number of rounds T.

7.2. AdaBoost and variants

As mentioned in Section 3, a simplified, non-adaptive version of AdaBoost can be derived as an instance of OS. To do this, we simply replace the loss function (Eq. (7)) in the binary boosting game of Section 3 with an exponential loss function $L(s) = e^{-\eta s}$ where $\eta > 0$ is a parameter of the game. As a special case of the discussion below, it will follow that

$$\phi_t(s) = \kappa^{T-t} e^{-\eta s}$$

where κ is the constant

$$\kappa = \frac{1-\delta}{2}e^{\eta} + \frac{1+\delta}{2}e^{-\eta}.$$

Also, the weight given to a chip at position s on round t is

$$\kappa^{T-t}\left(\frac{e^{\eta} - e^{-\eta}}{2}\right)e^{-\eta s}$$

which is proportional to $e^{-\eta s}$ (in other words, the weighting function is effectively unchanged from round to round). This weighting is the same as the one used by a non-adaptive version of AdaBoost in which all weak hypotheses are given equal weight. Since $e^{-\eta s}$ is an upper bound on the loss function of Eq. (7), Theorem 2 implies an upper bound on the fraction of mistakes of the final hypothesis of

$$\phi_0(0) = \kappa^T.$$

When

$$\eta = \frac{1}{2}\ln\left(\frac{1+\delta}{1-\delta}\right) \tag{28}$$

so that κ is minimized, this gives an upper bound of

$$(1-\delta^2)^{T/2} = (1-4\gamma^2)^{T/2}$$

which is equivalent to a non-adaptive version of Freund and Schapire's (1997) analysis.

We next consider a more general drifting game in n dimensions whose loss function is a sum of exponentials

$$L(\mathbf{s}) = \sum_{j=1}^{k} b_j \exp(-\eta_j \mathbf{u}_j \cdot \mathbf{s}) \tag{29}$$

where the b_j's, η_j's and $\mathbf{u}_j$'s are parameters with $b_j > 0$, $\eta_j > 0$, $\|\mathbf{u}_j\|_1 = 1$ and $\mathbf{u}_j \geq_\sigma \mathbf{0}$ for some sign vector σ. For this game, $B = [-1, +1]^n$ and $p = 1$. Many (non-adaptive) variants of AdaBoost correspond to special cases of this game. For instance, AdaBoost.M2 (Freund & Schapire, 1997), a multiclass version of AdaBoost, essentially uses the loss function

$$L(\mathbf{s}) = \sum_{j=2}^{n} e^{-(\eta/2)(s_1 - s_j)}$$

where we follow the multiclass setup of Section 3 so that n is the number of classes, and the first component in the drifting game is identified with the correct class. (As before, we

only consider a non-adaptive game in which $\eta > 0$ is a fixed, tunable parameter.) Likewise, AdaBoost.MH (Schapire & Singer, 1999), another multiclass version of AdaBoost, uses the loss function

$$L(\mathbf{s}) = e^{-\eta s_1} + \sum_{j=2}^{n} e^{\eta s_j}.$$

Note that both loss functions upper bound the "true" loss for multiclass boosting given in Eq. (11). Moreover, both functions clearly have the form given in Eq. (29).

We claim that, for the general game with loss function as in Eq. (29),

$$\phi_t(\mathbf{s}) = \sum_j b_j \kappa_j^{T-t} \exp(-\eta_j \mathbf{u}_j \cdot \mathbf{s}) \tag{30}$$

where

$$\kappa_j = \frac{1-\delta}{2} e^{\eta_j} + \frac{1+\delta}{2} e^{-\eta_j}.$$

Proof of Eq. (30) is by backwards induction on t. For fixed t and $\mathbf{s}$, let

$$\mathbf{w} = \sum_j b_j \kappa_j^{T-t} \left(\frac{e^{\eta_j} - e^{-\eta_j}}{2} \right) \mathbf{u}_j \exp(-\eta_j \mathbf{u}_j \cdot \mathbf{s}).$$

We will show that this is the minimizing weight vector that gets used by OS for a chip at position $\mathbf{s}$ at time t. Let

$$b'_j = b_j \kappa_j^{T-t} \exp(-\eta_j \mathbf{u}_j \cdot \mathbf{s}).$$

Note that

$$\begin{aligned} \phi_t(\mathbf{s}+\mathbf{z}) + \mathbf{w} \cdot \mathbf{z} &= \sum_j b'_j \left(\exp(-\eta_j \mathbf{u}_j \cdot \mathbf{z}) + \left(\frac{e^{\eta_j} - e^{-\eta_j}}{2} \right) \mathbf{u}_j \cdot \mathbf{z} \right) \\ &\le \sum_j b'_j \left(\frac{e^{\eta_j} + e^{-\eta_j}}{2} \right) \end{aligned} \tag{31}$$

since

$$e^{-\eta x} \le \left(\frac{e^{\eta} + e^{-\eta}}{2} \right) - \left(\frac{e^{\eta} - e^{-\eta}}{2} \right) x$$

for all $\eta \in \mathbb{R}$ and $x \in [-1, +1]$ by convexity of $e^{-\eta x}$. Also, by our assumptions on b_j, $\mathbf{u}_j$ and η_j, we can compute

$$\|\mathbf{w}\|_1 = \sum_j b'_j \left(\frac{e^{\eta_j} - e^{-\eta_j}}{2} \right). \tag{32}$$

Thus, combining Eqs. (31) and (32) gives

$$\begin{aligned}\phi_{t-1}(\mathbf{s}) &\le \sup_{\mathbf{z}\in B}(\phi_t(\mathbf{s}+\mathbf{z}) + \mathbf{w}\cdot\mathbf{z} - \delta\|\mathbf{w}\|_1)\\ &\le \sum_j b'_j\kappa_j\\ &= \sum_j b_j\kappa_j^{T-t+1}\ \exp(-\eta_j\mathbf{u}_j\cdot\mathbf{s}).\end{aligned}$$

This gives the needed upper bound on $\phi_{t-1}(\mathbf{s})$.

For the lower bound, using Theorem 7 (since L is unate with sign vector $-\boldsymbol{\sigma}$), we have

$$\begin{aligned}\phi_{t-1}(\mathbf{s}) &\ge \min_{\mathbf{w}\ge_\sigma \mathbf{0}}\max_{\mathbf{z}\in\{-\boldsymbol{\sigma},\,\boldsymbol{\sigma}\}}(\phi_t(\mathbf{s}+\mathbf{z}) + \mathbf{w}\cdot\mathbf{z} - \delta\|\mathbf{w}\|_1)\\ &= \min_{c\ge 0}\max\left\{\sum_j b'_j e^{-\eta_j} + c - \delta c, \sum_j b'_j e^{\eta_j} - c - \delta c\right\}\end{aligned}$$

where we have used $\mathbf{u}_j\cdot\boldsymbol{\sigma} = 1$ and $\mathbf{w}\cdot\boldsymbol{\sigma} = \|\mathbf{w}\|_1$ (since $\mathbf{u}_j \ge_\sigma \mathbf{0}$ and $\mathbf{w} \ge_\sigma \mathbf{0}$). We also have identified c with $\|\mathbf{w}\|_1$. Solving the min max expression gives the desired lower bound. This completes the proof of Eq. (30).

7.3. On-line learning algorithms

In this section, we show how Cesa-Bianchi et al.'s (1996) BW algorithm for combining expert advice can be derived as an instance of OS. We will also see how their algorithm can be generalized, and how Littlestone and Warmuth's (1994) weighted majority algorithm can also be derived and analyzed.

Suppose that we have access to m "experts." On each round t, each expert i provides a prediction $\xi_i^t \in \{-1, +1\}$. A "master" algorithm combines their predictions into its own prediction $\psi_t \in \{-1, +1\}$. An outcome $y_t \in \{-1, +1\}$ is then observed. The master makes a mistake if $\psi_t \ne y_t$, and similarly for expert i if $\xi_i^t \ne y_t$. The goal of the master is to minimize how many mistakes it makes relative to the best expert.

We will consider master algorithms which use a weighted majority vote to form their predictions; that is,

$$\psi_t = \text{sign}\left(\sum_{i=1}^m w_i^t\xi_i^t\right).$$

The problem is to derive a good choice of weights w_i^t. We also assume that the master algorithm is conservative in the sense that rounds on which the master's predictions are correct are effectively ignored (so that the weights w_i^t only depend upon previous rounds on which mistakes were made).

Let us suppose that there is one expert that makes at most k mistakes. We will (re)derive an algorithm (namely, BW) and a bound on the number of mistakes made by the master, given this assumption. Since we restrict our attention to conservative algorithms, we can assume without loss of generality that a mistake occurs on every round and simply proceed to bound the total number of rounds.

To set up the problem as a drifting game, we identify one chip with each of the experts. The problem is one dimensional so $n = 1$. The weights w_i^t selected by the master are the same as those chosen by the shepherd. Since we assume that the master makes a mistake on each round, we have for all t that

$$y_t \sum_i w_i^t \xi_i^t \leq 0. \tag{33}$$

Thus, if we define the drift z_i^t to be $-y_t \xi_i^t$, then

$$\sum_i w_i^t z_i^t \geq 0.$$

Setting $\delta = 0$, we see that Eq. (33) is equivalent to Eq. (1). Also, $B = \{-1, +1\}$.

Let M_i^t be the number of mistakes made by expert i on rounds $1, \ldots, t-1$. Then by definition of z_i^t,

$$s_i^t = 2M_i^t - t + 1.$$

Let the loss function L be

$$L(s) = \begin{cases} 1 & \text{if } s \leq 2k - T \\ 0 & \text{otherwise.} \end{cases} \tag{34}$$

Then $L(s_i^{T+1}) = 1$ if and only if expert i makes a total of k or fewer mistakes in T rounds. Thus, our assumption that the best expert makes at most k mistakes implies that

$$1 \leq \sum_i L\left(s_i^{T+1}\right). \tag{35}$$

On the other hand, Theorem 2 implies that

$$\frac{1}{m} \sum_i L\left(s_i^{T+1}\right) \leq \phi_0(0). \tag{36}$$

By an analysis similar to the one given in Section 7.1, it can be seen that

$$\phi_{t-1}(s) = \tfrac{1}{2}(\phi_t(s+1) + \phi_t(s-1)).$$

Solving this recurrence gives

$$\phi_t(s) = 2^{t-T}\binom{T-t}{\leq k - \frac{t+s}{2}}$$

where

$$\binom{n}{\leq k} = \sum_{i=0}^{k}\binom{n}{k}.$$

In particular,

$$\phi_0(0) = 2^{-T}\binom{T}{\leq k}. \tag{37}$$

Combining Eqs. (35)–(37) gives

$$\frac{1}{m} \leq 2^{-T}\binom{T}{\leq k}. \tag{38}$$

In other words, the number of mistakes T of the master algorithm must satisfy Eq. (38) and so must be at most

$$\max\left\{q \in \mathbb{N} : q \leq \lg m + \lg\binom{q}{\leq k}\right\},$$

the same bound given by Cesa-Bianchi et al. (1996).

The weighting function obtained is also equivalent to theirs since, by a similar argument to that used in Section 7.1, OS gives

$$\begin{aligned} w_i^t &= \frac{1}{2}\left(\phi_t\left(s_i^t - 1\right) - \phi_t\left(s_i^t + 1\right)\right) \\ &= 2^{t-T-1}\binom{T-t}{k - \frac{t+s-1}{2}} \\ &= 2^{t-T-1}\binom{T-t}{k - M_i^t}. \end{aligned}$$

Note that this argument can be generalized to the case in which the expert's predictions are not restricted to $\{-1, +1\}$ but instead may be all of $[-1, +1]$, or a subset of this interval, such as$\{-1, 0, +1\}$. The performance of each expert then is measured on each round using absolute loss $\frac{1}{2}\,|\xi_i^t - y_t|$ rather than whether or not it made a mistake. In this case, as in the analogous extension of boost-by-majority given in Section 3, we only need to replace B by $[-1, +1]$ or $\{-1, 0, +1\}$. The resulting bound on the number of mistakes of the master is

then the largest T for which $1/m \leq \phi_0(0)$ (note that $\phi_0(0)$ depends implicitly on T). The resulting master algorithm simply uses the weights computed by OS for the appropriate drifting game. It is an open problem to determine if this generalized algorithm enjoys strong optimality properties similar to those of BW (Cesa-Bianchi et al., 1996).

Littlestone and Warmuth's (1994) weighted majority algorithm can also be derived as an instance of OS. To do this, we simply replace the loss function L in the game above with

$$L(s) = \exp(-\eta(s - 2k + T))$$

for some parameter $\eta > 0$. This loss function upper bounds the one in Eq. (34). We assume that experts are permitted to output predictions in $[-1, +1]$ so that $B = [-1, +1]$. From the results of Section 7.2 applied to this drifting game,

$$\phi_t(s) = \kappa^{T-t} \exp(-\eta(s - 2k + T))$$

where

$$\kappa = \frac{e^{\eta} + e^{-\eta}}{2}.$$

Therefore, because one expert suffers loss at most k,

$$\frac{1}{m} \leq \phi_0(0) = \kappa^T e^{\eta(2k-T)}.$$

Equivalently, the number of mistakes T is at most

$$\frac{2\eta k + \ln m}{\ln\left(\frac{2}{1+e^{-2\eta}}\right)},$$

exactly the bound given by Littlestone and Warmuth (1994). The algorithm is also the same as theirs since the weight given to an expert (chip) at position s_i^t at time t is

$$w_i^t = \left(\frac{e^{\eta} - e^{-\eta}}{2}\right) \exp\bigl(-\eta\bigl(s_i^t - 2k + T\bigr)\bigr) \propto \exp\bigl(-2\eta M_i^t\bigr).$$

8. Open problems

This paper represents the first work on general drifting games. As such, there are many open problems.

We have presented closed-form solutions of the potential function for just a few special cases. Are there other cases in which such closed-form solutions are possible? In particular, can the boosting games of Section 3 corresponding to $B = \{-1, 0, +1\}$ and $B = [-1, +1]$ be put into closed-form?

For games in which a closed form is not possible, is there nevertheless a general method of characterizing the loss bound $\phi_0(\mathbf{0})$, say, as the number of rounds T gets large?

Side products of our work include new versions of boost-by-majority for the multiclass case, as well as binary cases in which the weak hypotheses have range $\{-1, 0, +1\}$ or $[-1, +1]$. However, the optimality proof for the drifting game only carries over to the boosting setting if the final hypothesis has the restricted forms given in Eqs. (4) and (10). Are the resulting boosting algorithms also optimal (for instance, in the sense proved by Freund (1995) for boost-by-majority) without these restrictions?

Likewise, can the extensions of the BW algorithm in Section 7.3 be shown to be optimal? Can this algorithm be extended using drifting games to the multiclass case, or to the case in which the master is allowed to output predictions in $[-1, +1]$ (suffering absolute loss)?

The OS algorithm is non-adaptive in the sense that δ must be known ahead of time. To what extent can OS be made adaptive? For instance, can Freund's (2001) recent technique for making boost-by-majority adaptive be carried over to the general drifting-game setting? Similarly, what happens if the number of rounds T is not known in advance?

Finally, are there other interesting drifting games for entirely different learning problems such as regression or density estimation?

Acknowledgments

Many thanks to Yoav Freund for very helpful discussions which led to this research.

Notes

1. In an earlier version of this paper, the "shepherd" was called the "drifter," a term that was found by some readers to be confusing. The name of the main algorithm has also been changed from "Shepherd" to "OS."
2. Of course, the real goal of a boosting algorithm is to find a hypothesis with low generalization error. In this paper, we only focus on the simplified problem of minimizing error on the given training examples.

References

Blackwell, D. (1956). An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics, 6:1*, 1–8.

Cesa-Bianchi, N., Freund, Y., Haussler, D., Helmbold, D. P., Schapire, R. E., & Warmuth, M. K. (1997). How to use expert advice. *Journal of the Association for Computing Machinery, 44:3*, 427–485.

Cesa-Bianchi, N., Freund, Y., Helmbold, D. P., & Warmuth, M. K. (1996). On-line prediction and conversion strategies. *Machine Learning, 25*, 71–110.

Freund, Y. (1995). Boosting a weak learning algorithm by majority. *Information and Computation, 121:2*, 256–285.

Freund, Y. (2001). An adaptive version of the boost by majority algorithm. *Machine Learning, 43:3*, 293–318.

Freund, Y. & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55:1*, 119–139.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association, 58:301*, 13–30.

Littlestone, N. & Warmuth, M. K. (1994). The weighted majority algorithm. *Information and Computation, 108*, 212–261.

Rockafellar, R. T. (1970). *Convex Analysis*. Princeton, NJ: Princeton University Press.
Schapire, R. E. & Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning, 37:3*, 297–336.

Received October 28, 1999
Revised October 28, 1999
Accepted June 1, 2000
Final manuscript July 31, 2000