Proceedings of the Ninth Annual Conference on Computational Learning Theory, 1996.

Game Theory, On-line Prediction and Boosting

Yoav Freund **Robert E. Schapire**

AT&T Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974-0636

$\{$yoav, schapire$\}$@research.att.com

Abstract

We study the close connections between game theory, on-line prediction and boosting. After a brief review of game theory, we describe an algorithm for learning to play repeated games based on the on-line prediction methods of Littlestone and Warmuth. The analysis of this algorithm yields a simple proof of von Neumann's famous minmax theorem, as well as a provable method of approximately solving a game. We then show that the on-line prediction model is obtained by applying this game-playing algorithm to an appropriate choice of game and that boosting is obtained by applying the same algorithm to the "dual" of this game.

1 INTRODUCTION

The purpose of this paper is to bring out the close connections between game theory, on-line prediction and boosting. Briefly, game theory is the study of games and other interactions of various sorts. On-line prediction is a learning model in which an agent predicts the classification of a sequence of items and attempts to minimize the total number of prediction errors. Finally, boosting is a method of converting a "weak" learning algorithm which performs only slightly better than random guessing into one that performs extremely well.

All three of these topics will be explained in more detail below. All have been studied extensively in the past. In this paper, the close relationship between these three seemingly unrelated topics will be brought out.

Here is an outline of the paper. We will begin with a review of game theory. Then we will describe an algorithm for learning to play repeated games based on the on-line prediction methods of Littlestone and Warmuth [15]. The analysis of this algorithm yields a new (as far as we know) and simple proof of von Neumann's famous minmax theorem, as well as a provable method of approximately solving a game.

In the last part of the paper we show that the on-line prediction model is obtained by applying the game-playing algorithm to an appropriate choice of game and that boosting is obtained by applying the same algorithm to the "dual" of this game.

2 GAME THEORY

We begin with a review of basic game theory. Further background can be found in any introductory text on game theory; see for instance Fudenberg and Tirole [11]. We study two-person games in normal form. That is, each game is defined by a matrix $\mathbf{M}$. There are two players called the row player and column player. To play the game, the row player chooses a row i, and, simultaneously, the column player chooses a column j. The selected entry $M(i,j)$ is the *loss* suffered by the row player.

For instance, the loss matrix for the children's game "Rock, Paper, Scissors" is given by:

	R	P	S
R	$\frac{1}{2}$	1	0
P	0	$\frac{1}{2}$	1
S	1	0	$\frac{1}{2}$

The row player's goal is to minimize its loss. Often, the goal of the column player is to maximize this loss, in which case the game is said to be "zero-sum." Most of our results are given in the context of a zero-sum game. However, our results also apply when no assumptions are made about the goal or strategy of the column player. We return to this point below.

For the sake of simplicity, we assume that all the losses are in the range $[0, 1]$. Simple scaling can be used to get more general results. Also, we restrict ourselves to the case where the number of choices available to each player is finite. However, most of the results translate with very mild additional assumptions to cases in which the number of choices is infinite. For a discussion of infinite matrix games see, for instance, Chapter 2 in Ferguson [3].

Home page: "http://www.research.att.com/orgs/ssr/people/uid". Expected to change to "http://www.research.att.com/~uid" sometime in the near future (for uid $\in$ $\{$yoav, schapire$\}$).

2.1 RANDOMIZED PLAY

As described above, the players choose a single row or column. Usually, this choice of play is allowed to be randomized. That is, the row player chooses a distribution $\mathbf{P}$ over the rows of $\mathbf{M}$, and (simultaneously) the column player chooses a distribution $\mathbf{Q}$ over columns. The row player's expected loss is easily computed as

$$\sum_{i,j} \mathbf{P}(i)\mathbf{M}(i,j)\mathbf{Q}(j) = \mathbf{P}^{\mathrm{T}}\mathbf{M}\mathbf{Q}.$$

For ease of notation, we will often denote this quantity by $\mathbf{M}(\mathbf{P},\mathbf{Q})$, and refer to it simply as the loss (rather than expected loss). In addition, if the row player chooses a distribution $\mathbf{P}$ but the column player chooses a single column j, then the (expected) loss is $\sum_i \mathbf{P}(i)\mathbf{M}(i,j)$ which we denote by $\mathbf{M}(\mathbf{P},j)$. The notation $\mathbf{M}(i,\mathbf{Q})$ is defined analogously.

Individual (deterministically chosen) rows i and columns j are called *pure strategies*. Randomized plays defined by distributions $\mathbf{P}$ and $\mathbf{Q}$ over rows and columns are called *mixed strategies*. The number of rows of the matrix $\mathbf{M}$ will be denoted by n.

2.2 SEQUENTIAL PLAY

Up until now, we have assumed that the players choose their (pure or mixed) strategies simultaneously. Suppose now that instead play is sequential. That is, suppose that the column player chooses its strategy $\mathbf{Q}$ *after* the row player has chosen and announced its strategy $\mathbf{P}$. Assume further that the column player's goal is to maximize the row player's loss (i.e., that the game is zero-sum). Then given $\mathbf{P}$, such a "worst-case" or "adversarial" column player will choose $\mathbf{Q}$ to maximize $\mathbf{M}(\mathbf{P},\mathbf{Q})$; that is, if the row player plays mixed strategy $\mathbf{P}$, then its payoff will be

$$\max_{\mathbf{Q}} \ \mathbf{M}(\mathbf{P},\mathbf{Q}). \qquad (1)$$

(It is understood here and throughout the paper that $\max_{\mathbf{Q}}$ denotes maximum over all probability distributions over columns; similarly, $\min_{\mathbf{P}}$ will always denote minimum over all probability distributions over rows. These extrema exist because the set of distributions over a finite space is compact.)

Knowing this, the row player should choose $\mathbf{P}$ to minimize Eq. (1), so the row player's loss will be

$$\min_{\mathbf{P}} \max_{\mathbf{Q}} \mathbf{M}(\mathbf{P},\mathbf{Q}).$$

A mixed strategy $\mathbf{P}^*$ realizing this minimum is called a *minmax strategy*.

Suppose now that the column player plays first and the row player can choose its play with the benefit of knowing the column player's chosen strategy $\mathbf{Q}$. Then by a symmetric argument, the loss of the row player will be

$$\max_{\mathbf{Q}} \min_{\mathbf{P}} \mathbf{M}(\mathbf{P},\mathbf{Q}),$$

and a $\mathbf{Q}^*$ realizing the maximum is called a *maxmin strategy*.

2.3 THE MINMAX THEOREM

Intuitively, we expect the player who chooses its strategy last to have the advantage since it plays knowing its opponent's strategy exactly. Thus, we expect

$$\max_{\mathbf{Q}} \min_{\mathbf{P}} \mathbf{M}(\mathbf{P},\mathbf{Q}) \leq \min_{\mathbf{P}} \max_{\mathbf{Q}} \mathbf{M}(\mathbf{P},\mathbf{Q}). \qquad (2)$$

We might go on naively to conjecture that the advantage of playing last is strict for some games so that, at least in some cases, the inequality in Eq. (2) is strict.

Surprisingly, it turns out not to matter which player plays first. Von Neumann's well-known minmax theorem states that the outcome is the same in either case so that

$$\max_{\mathbf{Q}} \min_{\mathbf{P}} \mathbf{M}(\mathbf{P},\mathbf{Q}) = \min_{\mathbf{P}} \max_{\mathbf{Q}} \mathbf{M}(\mathbf{P},\mathbf{Q}) \qquad (3)$$

for every matrix $\mathbf{M}$. The common value v of the two sides of the equality is called the *value* of the game $\mathbf{M}$. A proof of the minmax theorem will be given in Section 2.5.

In words, Eq. (3) means that the row player has a (minmax) strategy $\mathbf{P}^*$ such that regardless of the strategy $\mathbf{Q}$ played by the column player, the loss suffered $\mathbf{M}(\mathbf{P}^*,\mathbf{Q})$ will be *at most* v. Symmetrically, it means that the column player has a (maxmin) strategy $\mathbf{Q}^*$ such that, regardless of the strategy $\mathbf{P}$ played by the row player the loss will be *at least* v. This means that the strategies $\mathbf{Q}^*$ and $\mathbf{P}^*$ are optimal in a strong sense.

Thus, classical game theory says that given a (zero-sum) game $\mathbf{M}$, one should play using a minmax strategy. Such a strategy can be computed using linear programming.

However, there are a number of problems with this approach. For instance,

- $\mathbf{M}$ may be unknown;
- $\mathbf{M}$ may be so large that computing a minmax strategy using linear programming is infeasible;
- the column player may not be truly adversarial and may behave in a manner that admits loss significantly smaller than the game value v.

Overcoming these difficulties in the one-shot game is hopeless. But suppose instead that we are playing the game repeatedly. Then it is natural to ask if one can learn to play well against the particular opponent that is being faced.

2.4 REPEATED PLAY

Such a model of repeated play can be formalized as described below. To emphasize the roles of the two players, we refer to the row player as the *learner* and the column player as the *environment*.

Let $\mathbf{M}$ be a matrix, possibly unknown to the learner. The game is played repeatedly in a sequence of *rounds*. On round $t = 1,\ldots,T$:

1. the learner chooses mixed strategy $\mathbf{P}_t$;
2. the environment chooses mixed strategy $\mathbf{Q}_t$ (which may be chosen with knowledge of $\mathbf{P}_t$)
3. the learner is permitted to observe the loss $\mathbf{M}(i,\mathbf{Q}_t)$ for each row i; this is the loss it would have suffered had it played using pure strategy i;
4. the learner suffers loss $\mathbf{M}(\mathbf{P}_t,\mathbf{Q}_t)$.

The goal of the learner is to do almost as well as the best strategy against the actual sequence of plays $\mathbf{Q}_1,\ldots,\mathbf{Q}_T$

which were chosen by the environment. That is, the learner's goal is to suffer cumulative loss

$$\sum_{t=1}^{T} \mathbf{M}(\mathbf{P}_t, \mathbf{Q}_t)$$

which is "not much worse" than the loss of the *best* strategy in hindsight

$$\min_{\mathbf{P}} \sum_{t=1}^{T} \mathbf{M}(\mathbf{P}, \mathbf{Q}_t).$$

An algorithm for solving this problem can be derived by a direct generalization of Littlestone and Warmuth's "weighted majority algorithm" [15], and is essentially equivalent to our earlier "Hedge" algorithm [9]. The algorithm, called LW, is quite simple. The learner maintains nonnegative weights on the rows of $\mathbf{M}$; let $w_t(i)$ denote the weight at time t on row i. Initially, all the weights are set to unity: $w_1(i) = 1$. On each round t, the learner computes mixed strategy $\mathbf{P}_t$ by normalizing the weights:

$$\mathbf{P}_t(i) = \frac{w_t(i)}{\sum_i w_t(i)}.$$

Then, given $\mathbf{M}(i, \mathbf{Q}_t)$ for each i, the learner updates the weights by the simple multiplicative rule:

$$w_{t+1}(i) = w_t(i) \cdot \beta^{\mathbf{M}(i, \mathbf{Q}_t)}.$$

Here, $\beta \in [0, 1)$ is a parameter of the algorithm.

The main theorem concerning this algorithm is the following:

Theorem 1 *For any matrix* $\mathbf{M}$ *with* n *rows and entries in* $[0, 1]$*, and for any sequence of mixed strategies* $\mathbf{Q}_1, \ldots, \mathbf{Q}_T$ *played by the environment, the sequence of mixed strategies* $\mathbf{P}_1, \ldots, \mathbf{P}_T$ *produced by algorithm LW with parameter* $\beta \in [0, 1)$ *satisfy:*

$$\sum_{t=1}^{T} \mathbf{M}(\mathbf{P}_t, \mathbf{Q}_t) \le a_\beta \min_{\mathbf{P}} \sum_{t=1}^{T} \mathbf{M}(\mathbf{P}, \mathbf{Q}_t) + c_\beta \ln n$$

where

$$a_\beta = \frac{\ln(1/\beta)}{1-\beta} \qquad c_\beta = \frac{1}{1-\beta}.$$

Proof: The proof follows directly from Theorem 2 of Freund and Schapire [9], which in turn is a simple and direct generalization of Littlestone and Warmuth [15]. For completeness, we provide a short proof in the appendix. ■

As β approaches 1, a_β also approaches 1. In addition, for fixed β and as the number of rounds T becomes large, the second term $c_\beta \ln n$ becomes negligible (since it is fixed) relative to T. Thus, by choosing β close to 1, the learner can ensure that its loss will not be much worse than the loss of the best strategy. This is formalized in the following corollary:

Corollary 2 *Under the conditions of Theorem 1 and with* β *set to*

$$\frac{1}{1 + \sqrt{\frac{2 \ln n}{T}}},$$

the average per-trial loss suffered by the learner is

$$\frac{1}{T} \sum_{t=1}^{T} \mathbf{M}(\mathbf{P}_t, \mathbf{Q}_t) \le \min_{\mathbf{P}} \frac{1}{T} \sum_{t=1}^{T} \mathbf{M}(\mathbf{P}, \mathbf{Q}_t) + \Delta_T$$

where

$$\Delta_T = \sqrt{\frac{2 \ln n}{T}} + \frac{\ln n}{T} = O\left(\sqrt{\frac{\ln n}{T}}\right).$$

Proof: See Section 2.2 in Freund and Schapire [9]. ■

Since $\Delta_T \to 0$ as $T \to \infty$, we see that the amount by which the average per-trial loss of the learner exceeds that of the best mixed strategy can be made arbitrarily small for large T.

For simplicity, the results in the remainder of the paper are based on Corollary 2 rather than Theorem 1. The details of the algorithm about which this corollary applies are largely unimportant and could, in principle, be applied to any algorithm with similar properties. Indeed, algorithms for this problem with similar properties were derived by Hannan [13],[1] Blackwell [1] and Foster and Vohra [6, 5, 4]. Also, Fudenberg and Levine [10] independently proposed an algorithm equivalent to LW and proved a slightly weaker version of Corollary 2.

As a simple first corollary, we see that the loss of LW can never exceed the value of the game $\mathbf{M}$ by more than Δ_T.

Corollary 3 *Under the conditions of Corollary 2,*

$$\frac{1}{T} \sum_{t=1}^{T} \mathbf{M}(\mathbf{P}_t, \mathbf{Q}_t) \le v + \Delta_T$$

where v *is the value of the game* $\mathbf{M}$*.*

Proof: Let $\mathbf{P}^*$ be a minmax strategy for $\mathbf{M}$ so that for all column strategies $\mathbf{Q}$, $\mathbf{M}(\mathbf{P}^*, \mathbf{Q}) \le v$. Then, by Corollary 2,

$$\frac{1}{T} \sum_{t=1}^{T} \mathbf{M}(\mathbf{P}_t, \mathbf{Q}_t) \quad \le \quad \frac{1}{T} \sum_{t=1}^{T} \mathbf{M}(\mathbf{P}^*, \mathbf{Q}_t) + \Delta_T \le v + \Delta_T.$$

■

Note that in the analysis we made no assumption about the strategy used by the environment. Theorem 1 guarantees that its cumulative loss is not much larger than that of *any* fixed mixed strategy. As shown above, this implies, in particular, that the loss cannot be much larger than the game value. However, if the environment is non-adversarial, there might be a better fixed mixed strategy for the player, in which case the algorithm is guaranteed to be almost as good as this better strategy.

2.5 PROOF OF THE MINMAX THEOREM

More interestingly, Corollary 2 can be used to derive a very simple proof of von Neumann's minmax theorem. To prove this theorem, we need to show that

$$\min_{\mathbf{P}} \max_{\mathbf{Q}} \mathbf{M}(\mathbf{P}, \mathbf{Q}) \le \max_{\mathbf{Q}} \min_{\mathbf{P}} \mathbf{M}(\mathbf{P}, \mathbf{Q}). \tag{4}$$

[1] However, Hannan's algorithm requires prior knowledge of the *entire* game matrix.

(Proving that $\min_{\mathbf{P}} \max_{\mathbf{Q}} \mathbf{M}(\mathbf{P},\mathbf{Q}) \geq \max_{\mathbf{Q}} \min_{\mathbf{P}} \mathbf{M}(\mathbf{P},\mathbf{Q})$ is relatively straightforward and so is omitted.)

Suppose that we run algorithm LW against a maximally adversarial environment which always chooses strategies which maximize the learner's loss. That is, on each round t, the environment chooses

$$\mathbf{Q}_t = \arg\max_{\mathbf{Q}} \mathbf{M}(\mathbf{P}_t, \mathbf{Q}). \tag{5}$$

Let $\overline{\mathbf{P}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{P}_t$ and $\overline{\mathbf{Q}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{Q}_t$. Clearly, $\overline{\mathbf{P}}$ and $\overline{\mathbf{Q}}$ are probability distributions.

Then we have:

$$\begin{aligned}
&\min_{\mathbf{P}} \max_{\mathbf{Q}} \mathbf{P}^{\mathrm{T}}\mathbf{M}\mathbf{Q} \\
&\leq \max_{\mathbf{Q}} \overline{\mathbf{P}}^{\mathrm{T}}\mathbf{M}\mathbf{Q} \\
&= \max_{\mathbf{Q}} \frac{1}{T}\sum_{t=1}^{T} \mathbf{P}_t{}^{\mathrm{T}}\mathbf{M}\mathbf{Q} && \text{by definition of } \overline{\mathbf{P}} \\
&\leq \frac{1}{T}\sum_{t=1}^{T} \max_{\mathbf{Q}} \mathbf{P}_t{}^{\mathrm{T}}\mathbf{M}\mathbf{Q} \\
&= \frac{1}{T}\sum_{t=1}^{T} \mathbf{P}_t{}^{\mathrm{T}}\mathbf{M}\mathbf{Q}_t && \text{by definition of } \mathbf{Q}_t \\
&\leq \min_{\mathbf{P}} \frac{1}{T}\sum_{t=1}^{T} \mathbf{P}^{\mathrm{T}}\mathbf{M}\mathbf{Q}_t + \Delta_T && \text{by Corollary 2} \\
&= \min_{\mathbf{P}} \mathbf{P}^{\mathrm{T}}\mathbf{M}\overline{\mathbf{Q}} + \Delta_T && \text{by definition of } \overline{\mathbf{Q}} \\
&\leq \max_{\mathbf{Q}} \min_{\mathbf{P}} \mathbf{P}^{\mathrm{T}}\mathbf{M}\mathbf{Q} + \Delta_T\,.
\end{aligned}$$

Since Δ_T can be made arbitrarily close to zero, this proves Eq. (4) and the minmax theorem.

2.6 APPROXIMATELY SOLVING A GAME

Aside from yielding a proof for a famous theorem that by now has many proofs, the preceding derivation shows that algorithm LW can be used to find an approximate minmax or maxmin strategy. Finding these "optimal" strategies is called *solving* the game $\mathbf{M}$.

Skipping the first inequality of the sequence of equalities and inequalities above, we see that

$$\max_{\mathbf{Q}} \mathbf{M}(\overline{\mathbf{P}},\mathbf{Q}) \quad\leq\quad \max_{\mathbf{Q}} \min_{\mathbf{P}} \mathbf{M}(\mathbf{P},\mathbf{Q}) + \Delta_T = v + \Delta_T\,.$$

Thus, the vector $\overline{\mathbf{P}}$ is an approximate minmax strategy in the sense that for all column strategies $\mathbf{Q}$, $\mathbf{M}(\overline{\mathbf{P}},\mathbf{Q})$ does not exceed the game value v by more than Δ_T. Since Δ_T can be made arbitrarily small, this approximation can be made arbitrarily tight.

Similarly, ignoring the last inequality of this derivation, we have that

$$\min_{\mathbf{P}} \mathbf{M}(\mathbf{P},\overline{\mathbf{Q}}) \geq v - \Delta_T$$

so $\overline{\mathbf{Q}}$ also is an approximate maxmin strategy. Furthermore, it can be shown that $\mathbf{Q}_t$ satisfying Eq. (5) can always be chosen to be a pure strategy (i.e., a mixed strategy concentrated on a single column of $\mathbf{M}$). Therefore, the approximate maxmin strategy $\overline{\mathbf{Q}}$ has the additional favorable property of being *sparse* in the sense that at most T of its entries will be nonzero.

Viewing LW as a method of approximately solving a game will be central to our derivation of a boosting algorithm (Section 4).

Similar and closely related methods of approximately solving linear programming problems have previously appeared, for instance, in the work of Plotkin, Shmoys and Tardos [16].

3 ON-LINE PREDICTION

Since the game-playing algorithm LW presented in Section 2.4 is a direct generalization of the on-line prediction algorithm of Littlestone and Warmuth [15], it is not surprising that an on-line prediction algorithm can be derived from the more general game-playing algorithm by an appropriate choice of game $\mathbf{M}$. In this section, we make this connection explicit.

In the on-line prediction model, first introduced by Littlestone [14], the learner observes a sequence of examples and predicts their labels one at a time. The learner's goal is to minimize its prediction errors.

Formally, let X be a finite set of *instances*, and let $\mathcal{H}$ be a finite set of *hypotheses* $h : X \to \{0,1\}$. Let $c : X \to \{0,1\}$ be an unknown *target concept*, not necessarily in $\mathcal{H}$.[2]

In the on-line prediction model, learning takes place in a sequence of rounds. On round $t = 1,\ldots,T$:

1. the learner observes an example $x_t \in X$;
2. the learner makes a randomized prediction $\hat{y}_t \in \{0,1\}$ of the label associated with x_t;
3. the learner observes the correct label $c(x_t)$.

The goal of the learner is to minimize the expected number of mistakes that it makes relative to the best hypothesis in the space $\mathcal{H}$. (The expectation here is with respect to the learner's own randomization.) Thus, we ask that the learner perform well whenever the target c is "close" to one of the hypotheses in $\mathcal{H}$.

It is straightforward now to reduce the on-line prediction problem to a special case of the repeated game problem. The environment's choice of a column corresponds to a choice of an instance $x \in X$ that is presented to the learner on a given iteration. The learner's choice of a row corresponds to choosing a specific hypothesis $h \in \mathcal{H}$ and predicting the label $h(x)$. A mixed strategy for the learner corresponds to making a random choice of a hypothesis with which to predict. In this reduction the environment uses only pure strategies. The game matrix thus has $|\mathcal{H}|$ rows, indexed by $h \in \mathcal{H}$ and $|X|$ columns, indexed by $x \in X$. The matrix entry that is associated with hypothesis h and instance x is

$$\mathbf{M}(h,x) = \begin{cases} 1 & \text{if } h(x) \neq c(x) \\ 0 & \text{otherwise.} \end{cases}$$

[2] As was said above, much of this analysis can be generalized to infinite sets. The cardinality of the set of examples is actually of no real consequence. Littlestone and Warmuth [15] generalize their results to countably infinite sets of hypotheses, and Freund and Schapire [9] and Freund [8] give generalizations to uncountably infinite sets of hypotheses.

Thus, $\mathbf{M}(h,x)$ is 1 if and only if h disagrees with the target c on instance x. We call this a *mistake matrix*.

The application of the algorithm LW described in Section 2.4 to the on-line prediction problem is as follows.[3] We apply the algorithm to mistake matrix $\mathbf{M}$. On round t, given instance x_t, LW provides us with a distribution $\mathbf{P}_t$ over rows of $\mathbf{M}$ (i.e., over hypothesis space $\mathcal{H}$). We randomly select $h_t \in \mathcal{H}$ according to $\mathbf{P}_t$, and predict $\hat{y}_t = h_t(x_t)$. Next, given $c(x_t)$, we compute $\mathbf{M}(h, x_t)$ for each $h \in \mathcal{H}$ and update the weights maintained by LW. (Here, the strategy $\mathbf{Q}_t$ is simply the pure strategy concentrated on the x_t column of $\mathbf{M}$.)

For the analysis, note that

$$\begin{aligned}\mathbf{M}(\mathbf{P}_t, x_t) &= \sum_{h\in\mathcal{H}} \mathbf{P}_t(h)\mathbf{M}(h,x_t) \\ &= \Pr_{h\sim\mathbf{P}_t}\left[h(x_t) \neq c(x_t)\right] \\ &= \Pr\left[\hat{y}_t \neq c(x_t)\right].\end{aligned}$$

Therefore, the expected number of mistakes made by the learner equals

$$\sum_{t=1}^{T}\mathbf{M}(\mathbf{P}_t, x_t) \leq \min_{h\in\mathcal{H}}\sum_{t=1}^{T}\mathbf{M}(h,x_t) + O\left(\sqrt{T\ln|\mathcal{H}|}\right)$$

by a direct application of Corollary 2 (for an appropriate choice of β). Thus, the expected number of mistakes made by the learner cannot exceed the number of mistakes made by the best hypothesis in $\mathcal{H}$ by more than $O\left(\sqrt{T\ln|\mathcal{H}|}\right)$.

A more careful analysis (using Theorem 1 rather than Corollary 2) gives a better bound identical to that obtained by Littlestone and Warmuth [15] (not surprisingly). Still better bounds using more sophisticated methods were obtained by Cesa-Bianchi et al. [2] and Vovk [18].

This result can be straightforwardly generalized to any bounded loss function (such as square loss rather than zero-one mistake loss), and also to a setting in which the learner competes against a set of experts rather than a fixed set of hypotheses. (See, for instance, Cesa-Bianchi et al. [2] and Freund and Schapire [9].)

4 BOOSTING

The third topic of this paper is boosting. *Boosting* is the problem of converting a "weak" learning algorithm that performs just slightly better than random guessing into one that performs with arbitrarily good accuracy. The first provably effective boosting algorithm was discovered by Schapire [17]. Freund [7] subsequently presented a much improved boosting algorithm which is optimal in particular circumstances. The boosting algorithm derived in this section is closely related to Freund and Schapire's more recent "AdaBoost" boosting algorithm [9].

As in Section 3, let X be a space of instances, $\mathcal{H}$ a space of hypotheses, and c the target concept. For $\gamma > 0$, we say that algorithm WL is a γ-weak learning algorithm for $(\mathcal{H}, c)$ if, for any distribution $\mathbf{Q}$ over the set X, the algorithm takes as input a set of labeled examples distributed according to $\mathbf{Q}$ and outputs a hypothesis $h \in \mathcal{H}$ with error at most $1/2 - \gamma$, i.e., $\Pr_{x\sim\mathbf{Q}}\left[h(x) \neq c(x)\right] \leq \frac{1}{2} - \gamma$.

Given a weak learning algorithm, the goal of boosting is to run the weak learning algorithm many times on many distributions, and to combine the selected hypotheses into a final hypothesis with arbitrarily small error rate. For the purposes of this paper, we simplify the boosting model further to require that the final hypothesis have error zero so that all instances are correctly classified. The algorithm presented can certainly be modified to fit the more standard (and practical) model in which the final error must be less than some positive parameter ϵ (see Freund and Schapire [9] for more details).[4]

Thus, boosting proceeds in rounds. On round $t = 1, \ldots, T$:

1. the booster constructs a distribution D_t on X which is passed to the weak learner;
2. the weak learner produces a hypothesis $h_t \in \mathcal{H}$ with error at most $1/2 - \gamma$:
$$\Pr_{x\sim D_t}\left[h_t(x) \neq c(x)\right] \leq \tfrac{1}{2} - \gamma.$$

After T rounds, the weak hypotheses $h_1, \ldots, h_T$ are combined into a final hypothesis h_{fin}.

The important issues for designing a boosting algorithm are: (1) how to choose distributions D_t, and (2) how to combine the h_t's into a final hypothesis.

4.1 BOOSTING AND THE MINMAX THEOREM

Before describing our boosting algorithm, let us step back for a moment to consider the relationship between the mistake matrix $\mathbf{M}$ used in Section 3 and the minmax theorem. This relationship will turn out to be highly relevant to the design and understanding of the boosting algorithm.

Recall that the mistake matrix $\mathbf{M}$ has rows and columns indexed by hypotheses and instances, respectively, and that $\mathbf{M}(h,x) = 1$ if $h(x) \neq c(x)$ and is zero otherwise. Assuming $(\mathcal{H}, c)$ is γ-weakly learnable (so that there exists a γ-weak learning algorithm), what does the minmax theorem say about $\mathbf{M}$? Suppose that the value of $\mathbf{M}$ is v. Then

$$\begin{aligned}\min_{\mathbf{P}}\max_{x}\mathbf{M}(\mathbf{P},x) &= \min_{\mathbf{P}}\max_{\mathbf{Q}}\mathbf{M}(\mathbf{P},\mathbf{Q}) \\ &= v \\ &= \max_{\mathbf{Q}}\min_{\mathbf{P}}\mathbf{M}(\mathbf{P},\mathbf{Q}) \\ &= \max_{\mathbf{Q}}\min_{h}\mathbf{M}(h,\mathbf{Q}). \qquad (6)\end{aligned}$$

[3] The reduction is not specific to the use of LW. Other algorithms for playing repeated games can be combined with this reduction to give on-line learning algorithms. However, these algorithms need to be capable of working without complete knowledge of the matrix. It should be sufficient for the algorithm to receive as input only the identity and contents of columns that have been chosen by the environment in the past.

[4] The standard boosting model usually also includes a "confidence" parameter $\delta > 0$ which bounds the probability of the boosting algorithm failing to find a final hypothesis with low error. This parameter is necessary if we assume that the weak learner only succeeds *with high probability*. However, because we here make the simplifying assumption that the weak learner *always* succeeds in finding a weak hypothesis with error at most $1/2 - \gamma$, we have no need of a confidence parameter and instead require that the boosting algorithm succeed with absolute certainty.

(It is straightforward to show that, for any $\mathbf{Q}$, $\min_{\mathbf{P}} \mathbf{M}(\mathbf{P}, \mathbf{Q})$ is realized at a pure strategy h. Similarly for $\mathbf{P}$ and x.)

Note that, by $\mathbf{M}$'s definition,

$$\mathbf{M}(h, \mathbf{Q}) = \Pr_{x \sim \mathbf{Q}} \left[h(x) \neq c(x) \right].$$

Therefore, the right hand part of Eq. (6) says that there exists a distribution $\mathbf{Q}^*$ on X such that for every hypothesis h, $\mathbf{M}(h, \mathbf{Q}^*) = \Pr_{x \sim \mathbf{Q}^*} \left[h(x) \neq c(x) \right] \geq v$. However, because we assume γ-weak learnability, there must exist a hypothesis h such that

$$\Pr_{x \sim \mathbf{Q}^*} \left[h(x) \neq c(x) \right] \leq \tfrac{1}{2} - \gamma.$$

Combining these facts gives that $v \leq 1/2 - \gamma$.

On the other hand, the left part of Eq. (6) implies that there exists a distribution $\mathbf{P}^*$ over the hypothesis space $\mathcal{H}$ such that for every $x \in X$:

$$\mathbf{M}(\mathbf{P}^*, x) = \Pr_{h \sim \mathbf{P}^*} \left[h(x) \neq c(x) \right] \leq v \leq \tfrac{1}{2} - \gamma < \tfrac{1}{2}.$$

That is, every instance x is misclassified by less than $1/2$ of the hypotheses (as weighted by $\mathbf{P}^*$). Therefore, the target concept c is functionally equivalent to a weighted majority of hypotheses in $\mathcal{H}$.

To summarize this discussion, we have argued that if $(\mathcal{H}, c)$ are γ-weakly learnable, then c can be computed exactly as a weighted majority of hypotheses in $\mathcal{H}$. Moreover, the weights used in this function (defined by distribution $\mathbf{P}^*$ above) are not just any old weights, but rather are a minmax strategy for the game $\mathbf{M}$.

A similar proof technique was previously used by Goldmann, Håstad and Razborov [12] to prove a result about the representation power of circuits of weighted threshold gates.

4.2 IDEA FOR BOOSTING

The idea of our boosting algorithm then is to approximate c by approximating the weights of this function. Since these weights are a minmax strategy of the game $\mathbf{M}$, we might hope to apply the method described in Section 2.4 for approximately solving a game.

The problem is that the resulting algorithm does not fit the boosting model. Recall that on each round, algorithm LW computes a distribution over the rows of the game matrix (hypotheses, in the case of matrix $\mathbf{M}$). However, in the boosting model, we want to compute on each round a distribution over instances (columns of $\mathbf{M}$).

Since we have an algorithm which computes distributions over rows, but need one that computes distributions over columns, the obvious solution is to reverse the roles of rows and columns. This is exactly the approach that we follow. That is, rather than using game $\mathbf{M}$ directly, we construct the *dual* of $\mathbf{M}$ which is the identical game except that the roles of the row and column players have been reversed.

Constructing the dual $\mathbf{M}'$ of a game $\mathbf{M}$ is straightforward. First, we need to reverse row and column so we take the transpose $\mathbf{M}^{\mathrm{T}}$. This, however, is not enough since the column player of $\mathbf{M}$ wants to maximize the outcome, but the row player of $\mathbf{M}'$ wants to minimize the outcome (loss). Therefore, we also need to reverse the meaning of minimum and maximum which is easily done by negating the matrix yielding $-\mathbf{M}^{\mathrm{T}}$. Finally, to adhere to our convention of losses being in the range $[0, 1]$, we add the constant 1 to every outcome, which has no effect on the game. Thus, the dual $\mathbf{M}'$ of $\mathbf{M}$ is simply

$$\mathbf{M}' = \mathbf{1} - \mathbf{M}^{\mathrm{T}}$$

where **1** is an all 1's matrix of the appropriate dimensions.

In the case of the mistake matrix $\mathbf{M}$, the dual now has rows and columns indexed by instances and hypotheses, respectively, and each entry is

$$\mathbf{M}'(x, h) \;=\; 1 - \mathbf{M}(h, x) \;=\; \begin{cases} 1 & \text{if } h(x) = c(x) \\ 0 & \text{otherwise.} \end{cases}$$

Note that any minmax strategy of the game $\mathbf{M}$ becomes a maxmin strategy of the game $\mathbf{M}'$. Therefore, whereas before we were interested in finding an approximate minmax strategy of $\mathbf{M}$, we are now interested in finding an approximate maxmin strategy of $\mathbf{M}'$.

We can now apply algorithm LW to game matrix $\mathbf{M}'$ since, by the results of Section 2.6, this will lead to the construction of an approximate maxmin strategy. The reduction proceeds as follows: On round t of boosting

1. algorithm LW computes a distribution $\mathbf{P}_t$ over rows of $\mathbf{M}'$ (i.e., over X);

2. the boosting algorithm sets $D_t = \mathbf{P}_t$ and passes D_t to the weak learning algorithm;

3. the weak learner returns a hypothesis h_t satisfying
$$\Pr_{x \sim D_t} \left[h_t(x) = c(x) \right] \geq \tfrac{1}{2} + \gamma;$$

4. the weights maintained by algorithm LW are updated where $\mathbf{Q}_t$ is defined to be the pure strategy h_t.

According to the method of approximately solving a game given in Section 2.6, on each round t, $\mathbf{Q}_t$ may be a pure strategy h_t and should be chosen to maximize

$$\mathbf{M}'(\mathbf{P}_t, h_t) = \sum_x \mathbf{P}_t(x) \mathbf{M}'(x, h_t) = \Pr_{x \sim \mathbf{P}_t} \left[h_t(x) = c(x) \right].$$

In other words, h_t should have maximum accuracy with respect to distribution $\mathbf{P}_t$. This is exactly the goal of the weak learner. (Although it is not guaranteed to succeed in finding the *best* h_t, finding one of accuracy $1/2 + \gamma$ turns out to be sufficient for our purposes.)

Finally, this method suggests that $\overline{\mathbf{Q}} = (1/T)\sum_{t=1}^{T} \mathbf{Q}_t$ is an approximate maxmin strategy, and we know that the target c is equivalent to a majority of the hypotheses if weighted by a maxmin strategy of $\mathbf{M}'$. Since $\mathbf{Q}_t$ is in our case concentrated on pure strategy (hypothesis) h_t, this leads us to choose a final hypothesis h_{fin} which is the (simple) majority of $h_1, \ldots, h_T$.

4.3 ANALYSIS

Indeed, the resulting boosting procedure will compute a final hypothesis h_{fin} identical to c for sufficiently large T. We show in this section how this follows from Corollary 2.

As noted earlier, for all t,

$$\mathbf{M}'(\mathbf{P}_t, h_t) = \Pr_{x \sim \mathbf{P}_t} \left[h_t(x) = c(x) \right] \geq \tfrac{1}{2} + \gamma.$$

Therefore, by Corollary 2,

$$\tfrac{1}{2} + \gamma \le \frac{1}{T}\sum_{t=1}^{T}\mathbf{M}'(\mathbf{P}_t, h_t) \le \min_x \frac{1}{T}\sum_{t=1}^{T}\mathbf{M}'(x, h_t) + \Delta_T .$$

Therefore, for all x,

$$\frac{1}{T}\sum_{t=1}^{T}\mathbf{M}'(x, h_t) \ge \tfrac{1}{2} + \gamma - \Delta_T > \tfrac{1}{2} \qquad (7)$$

where the last inequality holds for sufficiently large T (specifically, when $\Delta_T < \gamma$). Note that, by definition of $\mathbf{M}'$, $\sum_{t=1}^{T}\mathbf{M}'(x, h_t)$ is exactly the number of hypotheses h_t which agree with c on instance x. Therefore, in words, Eq. (7) says that more than half the hypotheses h_t are correct on x. Therefore, by definition of h_{fin}, we have that $h_{\mathit{fin}}(x) = c(x)$ for all x.

For this to hold, we need only that $\Delta_T < \gamma$, which will be the case for $T = \Omega(\ln|X|/\gamma^2)$.

The resulting boosting algorithm, in which the game-playing subroutine LW has been "compiled out" is shown in Fig. 1. The algorithm is actually quite intuitive in this form: after each hypothesis h_t is observed, the weight associated with each instance x is decreased if h_t is correct on that instance and otherwise is increased. Thus, each distribution focuses on the examples most likely to be misclassified by the preceding hypotheses.

In practice, of course, the booster would not have access to the labels associated with the *entire* domain X. Rather, the booster would be given a labeled training set and all distributions would be computed over the training set. The generalization error of the final hypothesis can then be bounded using, for instance, standard "VC theory" (see Freund and Schapire [9] for more details).

A more sophisticated version of this algorithm, called AdaBoost, is given by Freund and Schapire [9]. The advantage of this version is that the learner does not need to know a priori the minimum accuracy rate of each weak hypothesis.

5 SUMMARY

In sum, we have shown how the two well-studied learning problems of on-line prediction and boosting can be cast in a single game-theoretic framework in which the two seemingly very different problems can be viewed as "duals" of one another.

We hope that the insight offered by this connection will help in the development and understanding of such learning algorithms since an algorithm for one problem may, in principle, be translated into an algorithm for the other. As a concrete example, the boosting algorithm described in this paper was derived from Littlestone and Warmuth's weighted majority algorithm by following this dual connection.

Acknowledgments

Thanks to Manfred Warmuth for many helpful discussions, and to Dean Foster and Rakesh Vohra for pointing us to relevant literature.

```
Input: instance space $X$ and target $c$
       $\gamma$-weak learning algorithm
Set $T = \left\lceil \frac{4}{\gamma^2} \ln|X| \right\rceil$ (so that $\Delta_T < \gamma$).
Set $\beta = 1/(1 + \sqrt{2\ln|X|/T})$.
Let $D_1(x) = 1/|X|$ for $x \in X$.
For $t = 1, \ldots, T$:
    • Pass distribution $D_t$ to weak learner.
    • Get back hypothesis $h_t$ such that
          $\Pr_{x \sim D_t}\left[h_t(x) \ne c(x)\right] \le \frac{1}{2} - \gamma$.
    • Update $D_t$:
          $D_{t+1}(x) = \frac{D_t(x)}{Z_t} \times \begin{cases} \beta & \text{if } h_t(x) = c(x) \\ 1 & \text{otherwise} \end{cases}$
      where $Z_t$ is a normalization constant (chosen so that
      $D_{t+1}$ will be a distribution).
Output final hypothesis $h_{\mathit{fin}} = \mathrm{MAJ}(h_1, \ldots, h_T)$.
```

Figure 1: The boosting algorithm.

References

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

[2] Nicolò Cesa-Bianchi, Yoav Freund, David P. Helmbold, David Haussler, Robert E. Schapire, and Manfred K. Warmuth. How to use expert advice. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 382–391, 1993.

[3] Thomas S. Ferguson. *Mathematical Statistics: A Decision Theoretic Approach*. Academic Press, 1967.

[4] Dean Foster and Rakesh Vohra. Regret in on-line decision making. unpublished manuscript, 1996.

[5] Dean Foster and Rakesh V. Vohra. Asymptotic calibration. unpublished manuscript, 1995.

[6] Dean P. Foster and Rakesh V. Vohra. A randomization rule for selecting forecasts. *Operations Research*, 41(4):704–709, July–August 1993.

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

[8] Yoav Freund. Predicting a binary sequence almost as well as the optimal biased coin. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, 1996.

[9] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory:*

Second European Conference, EuroCOLT '95, pages 23–37. Springer-Verlag, 1995. A draft of the journal version is available electronically (on our web pages, or by email request).

[10] Drew Fudenberg and David K. Levine. Consistency and cautious fictitious play. *Journal of Economic Dynamics and Control*, 19:1065–1089, 1995.

[11] Drew Fudenberg and Jean Tirole. *Game Theory*. MIT Press, 1991.

[12] Mikael Goldmann, Johan Håstad, and Alexander Razborov. Majority gates vs. general weighted threshold gates. *Computational Complexity*, 2:277–300, 1992.

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

[14] Nick Littlestone. Learning when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[15] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, 1994.

[16] Serge A. Plotkin, David B. Shmoys, and Éva Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20(2):257–301, May 1995.

[17] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[18] Volodimir G. Vovk. Aggregating strategies. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 371–383, 1990.

A PROOF OF THEOREM 1

For $t = 1, \ldots, T$, we have that

$$\begin{aligned}
\sum_{i=1}^{n} w_{t+1}(i) &= \sum_{i=1}^{n} w_t(i) \cdot \beta^{\mathbf{M}(i,\mathbf{Q}_t)} \\
&\leq \sum_{i=1}^{n} w_t(i) \cdot (1 - (1-\beta)\mathbf{M}(i,\mathbf{Q}_t)) \\
&= \left(\sum_{i=1}^{n} w_t(i)\right) \cdot (1 - (1-\beta)\mathbf{M}(\mathbf{P}_t,\mathbf{Q}_t)).
\end{aligned}$$

The first line uses the definition of $w_{t+1}(i)$. The second line follows from the fact that $\beta^x \leq 1 - (1-\beta)x$ for $\beta > 0$ and $x \in [0,1]$. The last line uses the definition of $\mathbf{P}_t$.

Unwrapping this simple recurrence gives

$$\sum_{i=1}^{n} w_{T+1}(i) \leq n \cdot \prod_{t=1}^{T} (1 - (1-\beta)\mathbf{M}(\mathbf{P}_t,\mathbf{Q}_t)). \tag{8}$$

(Recall that $w_1(i) = 1$.)

Next, note that, for any j,

$$\sum_{i=1}^{n} w_{T+1}(i) \geq w_{T+1}(j) = \beta^{\sum_{t=1}^{T} \mathbf{M}(j,\mathbf{Q}_t)}.$$

Combining with Eq. (8) and taking logs gives

$$\begin{aligned}
&(\ln\beta) \sum_{t=1}^{T} \mathbf{M}(j,\mathbf{Q}_t) \\
&\quad\leq \ln n + \sum_{t=1}^{T} \ln(1 - (1-\beta)\mathbf{M}(\mathbf{P}_t,\mathbf{Q}_t)) \\
&\quad\leq \ln n - (1-\beta) \sum_{t=1}^{T} \mathbf{M}(\mathbf{P}_t,\mathbf{Q}_t)
\end{aligned}$$

since $\ln(1-x) \leq -x$ for $x < 1$. Rearranging terms, and noting that this expression holds for any j gives

$$\sum_{t=1}^{T} \mathbf{M}(\mathbf{P}_t,\mathbf{Q}_t) \leq a_\beta \min_j \sum_{t=1}^{T} \mathbf{M}(j,\mathbf{Q}_t) + c_\beta \ln n.$$

Since the minimum (over mixed strategies $\mathbf{P}$) in the bound of the theorem must be achieved by a pure strategy j, this implies the theorem.