Journal of Machine Learning Research 14 (2013) 437-497 Submitted 8/11; Revised 1/13; Published 2/13

A Theory of Multiclass Boosting

Indraneel Mukherjee IMUKHERJ@CS.PRINCETON.EDU
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043, USA

Robert E. Schapire SCHAPIRE@CS.PRINCETON.EDU
Princeton University
Department of Computer Science
Princeton, NJ 08540 USA

Editor: Manfred Warmuth

Abstract

Boosting combines weak classifiers to form highly accurate predictors. Although the case of binary classification is well understood, in the multiclass setting, the "correct" requirements on the weak classifier, or the notion of the most efficient boosting algorithms are missing. In this paper, we create a broad and general framework, within which we make precise and identify the optimal requirements on the weak-classifier, as well as design the most effective, in a certain sense, boosting algorithms that assume such requirements.

Keywords: multiclass, boosting, weak learning condition, drifting games

1. Introduction

Boosting (Schapire and Freund, 2012) refers to a general technique of combining rules of thumb, or weak classifiers, to form highly accurate combined classifiers. Minimal demands are placed on the weak classifiers, so that a variety of learning algorithms, also called weak-learners, can be employed to discover these simple rules, making the algorithm widely applicable. The theory of boosting is well-developed for the case of binary classification. In particular, the exact requirements on the weak classifiers in this setting are known: any algorithm that predicts better than random on any distribution over the training set is said to satisfy the weak learning assumption. Further, boosting algorithms that minimize loss as efficiently as possible have been designed. Specifically, it is known that the Boost-by-majority (Freund, 1995) algorithm is optimal in a certain sense, and that AdaBoost (Freund and Schapire, 1997) is a practical approximation.

Such an understanding would be desirable in the multiclass setting as well, since many natural classification problems involve more than two labels, for example, recognizing a digit from its image, natural language processing tasks such as part-of-speech tagging, and object recognition in vision. However, for such multiclass problems, a complete theoretical understanding of boosting is lacking. In particular, we do not know the "correct" way to define the requirements on the weak classifiers, nor has the notion of optimal boosting been explored in the multiclass setting.

Straightforward extensions of the binary weak-learning condition to multiclass do not work. Requiring less error than random guessing on every distribution, as in the binary case, turns out to be too weak for boosting to be possible when there are more than two labels. On the other hand, requiring more than 50% accuracy even when the number of labels is much larger than two is too stringent, and simple weak classifiers like decision stumps fail to meet this criterion, even though they often can be combined to produce highly accurate classifiers (Freund and Schapire, 1996a). The most common approaches so far have relied on reductions to binary classification (Allwein et al., 2000), but it is hardly clear that the weak-learning conditions implicitly assumed by such reductions are the most appropriate.

The purpose of a weak-learning condition is to clarify the goal of the weak-learner, thus aiding in its design, while providing a specific minimal guarantee on performance that can be exploited by a boosting algorithm. These considerations may significantly impact learning and generalization because knowing the correct weak-learning conditions might allow the use of simpler weak classifiers, which in turn can help prevent overfitting. Furthermore, boosting algorithms that more efficiently and effectively minimize training error may prevent underfitting, which can also be important.

In this paper, we create a broad and general framework for studying multiclass boosting that formalizes the interaction between the boosting algorithm and the weak-learner. Unlike much, but not all, of the previous work on multiclass boosting, we focus specifically on the most natural, and perhaps weakest, case in which the weak classifiers are genuine classifiers in the sense of predicting a single multiclass label for each instance. Our new framework allows us to express a range of weak-learning conditions, both new ones and most of the ones that had previously been assumed (often only implicitly). Within this formalism, we can also now finally make precise what is meant by *correct* weak-learning conditions that are neither too weak nor too strong.

We focus particularly on a family of novel weak-learning conditions that have an especially appealing form: like the binary conditions, they require performance that is only slightly better than random guessing, though with respect to performance measures that are more general than ordinary classification error. We introduce a whole family of such conditions since there are many ways of randomly guessing on more than two labels, a key difference between the binary and multiclass settings. Although these conditions impose seemingly mild demands on the weak-learner, we show that each one of them is powerful enough to guarantee boostability, meaning that some combination of the weak classifiers has high accuracy. And while no individual member of the family is necessary for boostability, we also show that the entire family taken together is necessary in the sense that for every boostable learning problem, there exists one member of the family that is satisfied. Thus, we have identified a family of conditions which, as a whole, is necessary and sufficient for multiclass boosting. Moreover, we can combine the entire family into a single weak-learning condition that is necessary and sufficient by taking a kind of union, or logical OR, of all the members. This combined condition can also be expressed in our framework.

With this understanding, we are able to characterize previously studied weak-learning conditions. In particular, the condition implicitly used by AdaBoost.MH (Schapire and Singer, 1999), which is based on a one-against-all reduction to binary, turns out to be strictly stronger than necessary for boostability. This also applies to AdaBoost.M1 (Freund and Schapire, 1996a), the most direct generalization of AdaBoost to multiclass, whose conditions can be shown to be equivalent to those of AdaBoost.MH in our setting. On the other hand, the condition implicit to the SAMME algorithm by Zhu et al. (2009) is too weak in the sense that even when the condition is satisfied, no boosting algorithm can guarantee to drive down the training error. Finally, the condition implicit to AdaBoost.MR (Schapire and Singer, 1999; Freund and Schapire, 1996a) (also called AdaBoost.M2) turns out to be exactly necessary and sufficient for boostability.

Employing proper weak-learning conditions is important, but we also need boosting algorithms that can exploit these conditions to effectively drive down error. For a given weak-learning condition, the boosting algorithm that drives down training error most efficiently in our framework can be understood as the optimal strategy for playing a certain two-player game. These games are non-trivial to analyze. However, using the powerful machinery of drifting games (Freund and Opper, 2002; Schapire, 2001), we are able to compute the optimal strategy for the games arising out of each weak-learning condition in the family described above. Compared to earlier work, our optimality results hold more generally and also achieve tighter bounds. These optimal strategies have a natural interpretation in terms of random walks, a phenomenon that has been observed in other settings (Abernethy et al., 2008; Freund, 1995).

We also analyze the optimal boosting strategy when using the minimal weak learning condition, and this poses additional challenges. Firstly, the minimal weak learning condition has multiple natural formulations—for example, as the union of all the conditions in the family described above, or the formulation used in AdaBoost.MR—and each formulation leading to a different game specification. A priori, it is not clear which game would lead to the best strategy. We resolve this dilemma by proving that the optimal strategies arising out of different formulations of the same weak learning condition lead to algorithms that are essentially equally good, and therefore we are free to choose whichever formulation leads to an easier analysis without fear of suffering in performance. We choose the union of conditions formulation, since it leads to strategies that share the same interpretation in terms of random walks as before. However, even with this choice, the resulting games are hard to analyze, and although we can explicitly compute the optimum strategies in general, the computational complexity is usually exponential in the number of classes. Nevertheless, we identify key situations under which efficient computation is possible.

The game-theoretic strategies are non-adaptive in that they presume prior knowledge about the *edge*, that is, how much better than random are the weak classifiers. Algorithms that are adaptive, such as AdaBoost, are much more practical because they do not require such prior information. We show therefore how to derive an adaptive boosting algorithm by modifying the game-theoretic strategy based on the minimal condition. This algorithm enjoys a number of theoretical guarantees. Unlike some of the non-adaptive strategies, it is efficiently computable, and since it is based on the minimal weak learning condition, it makes minimal assumptions. In fact, whenever presented with a boostable learning problem, this algorithm can approach zero training error at an exponential rate. More importantly, the algorithm is effective even beyond the boostability framework. In particular, we show empirical consistency, that is, the algorithm always converges to the minimum of a certain exponential loss over the training data, whether or not the data set is boostable. Furthermore, using the results in Mukherjee et al. (2011) we can show that this convergence occurs rapidly.

Our focus in this paper is only on minimizing training error, which, for the algorithms we derive, provably decreases exponentially fast with the number of rounds of boosting under boostability assumptions. Such results can be used in turn to derive bounds on the generalization error using standard techniques that have been applied to other boosting algorithms (Schapire et al., 1998; Freund and Schapire, 1997; Koltchinskii and Panchenko, 2002). Consistency in the multiclass classification setting has been studied by Tewari and Bartlett (2007) and has been shown to be trickier than binary classification consistency. Nonetheless, by following the approach in Bartlett and Traskin (2007) for showing consistency in the binary setting, we are able to extend the empirical consistency guarantees to general consistency guarantees in the multiclass setting: we show that under certain conditions and with sufficient data, our adaptive algorithm approaches the Bayes-optimum error on the *test* data set.

We present experiments aimed at testing the efficacy of the adaptive algorithm when working with a very weak weak-learner to check that the conditions we have identified are indeed weaker than others that had previously been used. We find that our new adaptive strategy achieves low test error compared to other multiclass boosting algorithms which usually heavily underfit. This validates the potential practical benefit of a better theoretical understanding of multiclass boosting.

1.1 Previous Work

The first boosting algorithms were given by Schapire (1990) and Freund (1995), followed by their AdaBoost algorithm (Freund and Schapire, 1997). Multiclass boosting techniques include AdaBoost.M1 and AdaBoost.M2 (Freund and Schapire, 1997), as well as AdaBoost.MH and AdaBoost.MR (Schapire and Singer, 1999). Other approaches include the work by Eibl and Pfeiffer (2005) and Zhu et al. (2009). There are also more general approaches that can be applied to boosting including Allwein et al. (2000), Beygelzimer et al. (2009), Dietterich and Bakiri (1995), Hastie and Tibshirani (1998) and Li (2010). Two game-theoretic perspectives have been applied to boosting. The first one (Freund and Schapire, 1996b; Rätsch and Warmuth, 2005) views the weak-learning condition as a minimax game, while drifting games (Schapire, 2001; Freund, 1995) were designed to analyze the most efficient boosting algorithms. These games have been further analyzed in the multiclass and continuous time setting in Freund and Opper (2002).

2. Framework

We introduce some notation. Unless otherwise stated, matrices will be denoted by bold capital letters like $\mathbf{M}$, and vectors by bold small letters like $\mathbf{v}$. Entries of a matrix and vector will be denoted as $M(i,j)$ or $v(i)$, while $\mathbf{M}(i)$ will denote the ith row of a matrix. The inner product of two vectors $\mathbf{u},\mathbf{v}$ is denoted by $\langle \mathbf{u},\mathbf{v}\rangle$. The Frobenius inner product $\mathrm{Tr}\left(\mathbf{A}\mathbf{B}'\right)$ of two matrices $\mathbf{A},\mathbf{B}$ will be denoted by $\mathbf{A}\bullet\mathbf{B}'$, where $\mathbf{B}'$ is the transpose of $\mathbf{B}$. The indicator function is denoted by $\mathbf{1}\left[\cdot\right]$. The set of all distributions over the set $\{1,\ldots,k\}$ will be denoted by $\Delta\{1,\ldots,k\}$, and in general, the set of all distributions over any set S will be denoted by $\Delta(S)$.

In multiclass classification, we want to predict the labels of examples lying in some set X. We are provided a training set of labeled examples $\{(x_1,y_1),\ldots,(x_m,y_m)\}$, where each example $x_i \in X$ has a label y_i in the set $\{1,\ldots,k\}$.

Boosting combines several mildly powerful predictors, called *weak classifiers*, to form a highly accurate combined classifier, and has been previously applied for multiclass classification. In this paper, we only allow weak classifiers that predict a single class for each example. This is appealing, since the combined classifier has the same form, although it differs from what has been used in much previous work. Later we will expand our framework to include *multilabel* weak classifiers, that may predict multiple labels per example.

We adopt a game-theoretic view of boosting. A game is played between two players, Booster and Weak-Learner, for a fixed number of rounds T. With binary labels, Booster outputs a distribution in each round, and Weak-Learner returns a weak classifier achieving more than 50% accuracy on that distribution. The multiclass game is an extension of the binary game. In particular, in each round t:

- Booster creates a cost-matrix $\mathbf{C}_t \in \mathbb{R}^{m\times k}$, specifying to Weak-Learner that the cost of classifying example x_i as l is $C_t(i,l)$. The cost-matrix may not be arbitrary, but should conform to certain restrictions as discussed below.

- Weak-Learner returns some weak classifier $h_t\colon X \to \{1,\ldots,k\}$ from a fixed space $h_t \in \mathcal{H}$ so that the cost incurred is
$$\mathbf{C}_t \bullet \mathbf{1}_{h_t} = \sum_{i=1}^{m} C_t(i, h_t(x_i)),$$
is "small enough", according to some conditions discussed below. Here by $\mathbf{1}_h$ we mean the $m \times k$ matrix whose (i,j)-th entry is $\mathbf{1}\left[h(i) = j\right]$.

- Booster computes a weight α_t for the current weak classifier based on how much cost was incurred in this round.

At the end, Booster predicts according to the weighted plurality vote of the classifiers returned in each round:

$$H(x) \triangleq \operatorname*{argmax}_{l\in\{1,\ldots,k\}} f_T(x,l), \text{ where } f_T(x,l) \triangleq \sum_{t=1}^{T} \mathbf{1}\left[h_t(x) = l\right]\alpha_t. \tag{1}$$

By carefully choosing the cost matrices in each round, Booster aims to minimize the training error of the final classifier H, even when Weak-Learner is adversarial. The restrictions on cost-matrices created by Booster, and the maximum cost Weak-Learner can suffer in each round, together define the *weak-learning condition* being used. For binary labels, the traditional weak-learning condition states: for any non-negative weights $w(1),\ldots,w(m)$ on the training set, the error of the weak classifier returned is at most $(1/2-\gamma/2)\sum_i w_i$. Here γ parametrizes the condition. There are many ways to translate this condition into our language. The one with fewest restrictions on the cost-matrices requires labeling correctly should be less costly than labeling incorrectly:

$$\forall i : C(i,y_i) \le C(i,\bar{y}_i) \text{ (here } \bar{y}_i \neq y_i \text{ is the other binary label)},$$

while the restriction on the returned weak classifier h requires less cost than predicting randomly:

$$\sum_i C(i,h(x_i)) \le \sum_i \left\{ \left(\frac{1}{2}-\frac{\gamma}{2}\right) C(i,\bar{y}_i) + \left(\frac{1}{2}+\frac{\gamma}{2}\right) C(i,y_i) \right\}.$$

By the correspondence $w(i) = C(i,\bar{y}_i) - C(i,y_i)$, we may verify the two conditions are the same.

We will rewrite this condition after making some simplifying assumptions. Henceforth, without loss of generality, we assume that the true label is always 1. Let $\mathcal{C}^{\text{bin}} \subseteq \mathbb{R}^{m\times 2}$ consist of matrices $\mathbf{C}$ which satisfy $C(i,1) \le C(i,2)$. Further, let $\mathbf{U}^{\text{bin}}_{\gamma} \in \mathbb{R}^{m\times 2}$ be the matrix whose each row is $(1/2+\gamma/2, 1/2-\gamma/2)$. Then, Weak-Learner searching space $\mathcal{H}$ satisfies the binary weak-learning condition if: $\forall \mathbf{C} \in \mathcal{C}^{\text{bin}}, \exists h \in \mathcal{H} : \mathbf{C} \bullet \left(\mathbf{1}_h - \mathbf{U}^{\text{bin}}_{\gamma}\right) \le \mathbf{0}$. There are two main benefits to this reformulation. With linear homogeneous constraints, the mathematics is simplified, as will be apparent later. More importantly, by varying the restrictions $\mathcal{C}^{\text{bin}}$ on the cost vectors and the matrix $\mathbf{U}^{\text{bin}}$, we can generate a vast variety of weak-learning conditions for the multiclass setting $k \ge 2$ as we now show.

Let $\mathcal{C} \subseteq \mathbb{R}^{m\times k}$ and let $\mathbf{B} \in \mathbb{R}^{m\times k}$ be a matrix which we call the *baseline*. We say a weak classifier space $\mathcal{H}$ satisfies the condition $(\mathcal{C}, \mathbf{B})$ if

$$\forall \mathbf{C} \in \mathcal{C}, \exists h \in \mathcal{H} : \quad \mathbf{C} \bullet (\mathbf{1}_h - \mathbf{B}) \leq \mathbf{0}, \quad \text{i.e.,} \sum_{i=1}^{m} C(i, h(i)) \leq \sum_{i=1}^{m} \langle \mathbf{C}(i), \mathbf{B}(i) \rangle . \tag{2}$$

In (2), the variable matrix $\mathbf{C}$ specifies how costly each misclassification is, while the baseline $\mathbf{B}$ specifies a weight for each misclassification. The condition therefore states that a weak classifier should not exceed the average cost when weighted according to baseline $\mathbf{B}$. This large class of weak-learning conditions captures many previously used conditions, such as the ones used by AdaBoost.M1 (Freund and Schapire, 1996a), AdaBoost.MH (Schapire and Singer, 1999) and AdaBoost.MR (Freund and Schapire, 1996a; Schapire and Singer, 1999) (see below), as well as novel conditions introduced in the next section.

By studying this vast class of weak-learning conditions, we hope to find the one that will serve the main purpose of the boosting game: finding a convex combination of weak classifiers that has zero training error. For this to be possible, at the minimum the weak classifiers should be sufficiently rich for such a perfect combination to exist. Formally, a collection $\mathcal{H}$ of weak classifiers is *boostable* if it is eligible for boosting in the sense that there exists a weighting $\boldsymbol{\lambda}$ on the votes forming a distribution that linearly separates the data: $\forall i : \operatorname{argmax}_{l\in\{1,\ldots,k\}} \sum_{h\in\mathcal{H}} \lambda(h) \mathbf{1}[h(x_i) = l] = y_i$. The weak-learning condition plays two roles. It rejects spaces that are not boostable, and provides an algorithmic means of searching for the right combination. Ideally, the second factor will not cause the weak-learning condition to impose additional restrictions on the weak classifiers; in that case, the weak-learning condition is merely a reformulation of being boostable that is more appropriate for deriving an algorithm. In general, it could be *too strong*, that is, certain boostable spaces will fail to satisfy the conditions. Or it could be *too weak*, that is, non-boostable spaces might satisfy such a condition. Booster strategies relying on either of these conditions will fail to drive down error, the former due to underfitting, and the latter due to overfitting. Later we will describe conditions captured by our framework that avoid being too weak or too strong. But before that, we show in the next section how our flexible framework captures weak learning conditions that have appeared previously in the literature.

3. Old Conditions

In this section, we rewrite, in the language of our framework, the weak learning conditions explicitly or implicitly employed in the multiclass boosting algorithms SAMME (Zhu et al., 2009), AdaBoost.M1 (Freund and Schapire, 1996a), and AdaBoost.MH and AdaBoost.MR (Schapire and Singer, 1999). This will be useful later on for comparing the strengths and weaknesses of the various conditions. We will end this section with a curious equivalence between the conditions of AdaBoost.MH and AdaBoost.M1.

Recall that we have assumed the correct label is 1 for every example. Nevertheless, we continue to use y_i to denote the correct label in this section.

3.1 Old Conditions in the New Framework

Here we restate, in the language of our new framework, the weak learning conditions of four algorithms that have earlier appeared in the literature.

3.1.1 SAMME

The SAMME algorithm (Zhu et al., 2009) requires less error than random guessing on any distribution on the examples. Formally, a space $\mathcal{H}$ satisfies the condition if there is a $\gamma' > 0$ such that,

$$\forall d(1),\ldots,d(m) \geq 0, \exists h \in \mathcal{H} : \sum_{i=1}^{m} d(i)\mathbf{1}\left[h(x_i) \neq y_i\right] \leq (1 - 1/k - \gamma') \sum_{i=1}^{m} d(i). \tag{3}$$

Define a cost matrix $\mathbf{C}$ whose entries are given by

$$C(i,j) = \begin{cases} d(i) & \text{if } j \neq y_i, \\ 0 & \text{if } j = y_i. \end{cases}$$

Then the left hand side of (3) can be written as

$$\sum_{i=1}^{m} C(i, h(x_i)) = \mathbf{C} \bullet \mathbf{1}_h.$$

Next let $\gamma = (k/(k-1))\gamma'$ and define baseline $\mathbf{U}_\gamma$ to be the multiclass extension of $\mathbf{U}^{\text{bin}}$,

$$U_\gamma(i,l) = \begin{cases} \frac{(1-\gamma)}{k} + \gamma & \text{if } l = y_i, \\ \frac{(1-\gamma)}{k} & \text{if } l \neq y_i. \end{cases}$$

Then notice that for each i, we have

$$\begin{aligned} \left\langle \mathbf{C}(i), \mathbf{U}_\gamma(i) \right\rangle &= \sum_{l \neq y_i} C(i,l) U_\gamma(i,l) \\ &= (k-1)\frac{(1-\gamma)}{k} d(i) \\ &= \left(1 - \frac{1}{k} - \left(1 - \frac{1}{k}\right)\gamma\right) d(i) \\ &= \left(1 - \frac{1}{k} - \gamma'\right) d(i). \end{aligned}$$

Therefore, the right hand side of (3) can be written as

$$\sum_{i=1}^{m} \sum_{l \neq y_i} C(i,l) U_\gamma(i,l) = \mathbf{C} \bullet \mathbf{U}_\gamma,$$

since $C(i, y_i) = 0$ for every example i. Define $\mathcal{C}^{\text{SAM}}$ to be the following collection of cost matrices:

$$\mathcal{C}^{\text{SAM}} \triangleq \left\{ \mathbf{C} : C(i,l) = \begin{cases} 0 & \text{if } l = y_i, \\ t_i & \text{if } l \neq y_i, \end{cases} \text{for non-negative } t_1, \ldots, t_m. \right\}$$

Using the last two equations, (3) is equivalent to

$$\forall \mathbf{C} \in \mathcal{C}^{\text{SAM}}, \exists h \in \mathcal{H} : \mathbf{C} \bullet \left(\mathbf{1}_h - \mathbf{U}_\gamma\right) \leq 0.$$

Therefore, the weak-learning condition of SAMME is given by $(\mathcal{C}^{\text{SAM}}, \mathbf{U}_\gamma)$.

3.1.2 ADABOOST.M1

AdaBoost.M1 (Freund and Schapire, 1997) measures the performance of weak classifiers using ordinary error. It requires $1/2+\gamma/2$ accuracy with respect to any non-negative weights $d(1),\ldots,d(m)$ on the training set:

$$\begin{aligned}\sum_{i=1}^m d(i)\mathbf{1}\left[h(x_i)\neq y_i\right] &\leq (1/2-\gamma/2)\sum_{i=1}^m d(i), \\ \text{i.e., } \sum_{i=1}^m d(i)[\![h(x_i)\neq y_i]\!] &\leq -\gamma\sum_{i=1}^m d(i).\end{aligned} \tag{4}$$

where $[\![\cdot]\!]$ is the ±1 indicator function, taking value $+1$ when its argument is true, and -1 when false. Using the transformation

$$C(i,l)=[\![l\neq y_i]\!]d(i)$$

we may rewrite (4) as

$$\begin{gathered}\forall C\in\mathbb{R}^{m\times k}\text{ satisfying }0\leq -C(i,y_i)=C(i,l)\text{ for }l\neq y_i, \\ \exists h\in\mathcal{H}:\sum_{i=1}^m C(i,h(x_i))\leq\gamma\sum_{i=1}^m C(i,y_i) \\ \text{i.e.,}\quad \forall\mathbf{C}\in\mathcal{C}^{\mathrm{M1}},\exists h\in\mathcal{H}:\mathbf{C}\bullet\left(\mathbf{1}_h-\mathbf{B}_\gamma^{\mathrm{M1}}\right)\leq 0,\end{gathered} \tag{5}$$

where $\mathbf{B}_\gamma^{\mathrm{M1}}(i,l)=\gamma\mathbf{1}\left[l=y_i\right]$, and $\mathcal{C}^{\mathrm{M1}}\subseteq\mathbb{R}^{m\times k}$ consists of matrices satisfying the constraints in (5).

3.1.3 ADABOOST.MH

AdaBoost.MH (Schapire and Singer, 1999) is a popular multiclass boosting algorithm that is based on the one-against-all reduction, and was originally designed to use weak-hypotheses that return a prediction for every example and every label. The implicit weak learning condition requires that for any matrix with non-negative entries $d(i,l)$, the weak-hypothesis should achieve $1/2+\gamma$ accuracy

$$\sum_{i=1}^m\left\{\mathbf{1}\left[h(x_i)\neq y_i\right]d(i,y_i)+\sum_{l\neq y_i}\mathbf{1}\left[h(x_i)=l\right]d(i,l)\right\}\quad\leq\quad\left(\frac{1}{2}-\frac{\gamma}{2}\right)\sum_{i=1}^m\sum_{l=1}^k d(i,l). \tag{6}$$

This can be rewritten as

$$\begin{aligned}&\sum_{i=1}^m\left\{-\mathbf{1}\left[h(x_i)=y_i\right]d(i,y_i)+\sum_{l\neq y_i}\mathbf{1}\left[h(x_i)=l\right]d(i,l)\right\} \\ &\quad\leq\sum_{i=1}^m\left\{\left(\frac{1}{2}-\frac{\gamma}{2}\right)\sum_{l\neq y_i}d(i,l)-\left(\frac{1}{2}+\frac{\gamma}{2}\right)d(i,y_i)\right\}.\end{aligned}$$

Using the mapping

$$C(i,l)=\begin{cases}d(i,l) & \text{if } l\neq y_i\\ -d(i,l) & \text{if } l=y_i,\end{cases}$$

their weak-learning condition may be rewritten as follows

$$\begin{aligned}
&\forall \mathbf{C} \in \mathbb{R}^{m\times k} \text{ satisfying } C(i,y_i) \le 0, C(i,l) \ge 0 \text{ for } l \ne y_i, \\
&\quad \exists h \in \mathcal{H} : \\
&\quad sum_{i=1}^{m} C(i,h(x_i)) \le \sum_{i=1}^{m} \left\{ \left(\frac{1}{2}+\frac{\gamma}{2}\right) C(i,y_i) + \left(\frac{1}{2}-\frac{\gamma}{2}\right) \sum_{l\ne y_i} C(i,l) \right\}.
\end{aligned} \tag{7}$$

Defining $\mathcal{C}^{\mathrm{MH}}$ to be the space of all cost matrices satisfying the constraints in (7), the above condition is the same as

$$\forall \mathbf{C} \in \mathcal{C}^{\mathrm{MH}}, \exists h \in \mathcal{H} : \mathbf{C} \bullet \left(\mathbf{1}_h - \mathbf{B}_\gamma^{\mathrm{MH}}\right) \le 0,$$

where $\mathbf{B}_\gamma^{\mathrm{MH}}(i,l) = (1/2 + \gamma[\![l = y_i]\!]/2)$.

3.1.4 ADABOOST.MR

AdaBoost.MR (Schapire and Singer, 1999) is based on the all-pairs multiclass to binary reduction. Like AdaBoost.MH, it was originally designed to use weak-hypotheses that return a prediction for every example and every label. The weak learning condition for AdaBoost.MR requires that for any non-negative cost-vectors $\{d(i,l)\}_{l\ne y_i}$, the weak-hypothesis returned should satisfy the following:

$$\begin{aligned}
\sum_{i=1}^{m} \sum_{l\ne y_i} \left(\mathbf{1}[h(x_i)=l] - \mathbf{1}[h(x_i)=y_i]\right) d(i,l) &\le -\gamma \sum_{i=1}^{m} \sum_{l\ne y_i} d(i,l) \\
\text{i.e., } \sum_{i=1}^{m} \left\{ -\mathbf{1}[h(x_i)=y_i] \sum_{l\ne y_i} d(i,l) + \sum_{l\ne y_i} \mathbf{1}[h(x_i)=l]\, d(i,l) \right\} &\le -\gamma \sum_{i=1}^{m} \sum_{l\ne y_i} d(i,l).
\end{aligned}$$

Substituting

$$C(i,l) = \begin{cases} d(i,l) & l \ne y_i \\ -\sum_{l\ne y_i} d(i,l) & l = y_i, \end{cases}$$

we may rewrite AdaBoost.MR's weak-learning condition as

$$\begin{aligned}
&\forall \mathbf{C} \in \mathbb{R}^{m\times k} \text{ satisfying } C(i,l) \ge 0 \text{ for } l \ne y_i, C(i,y_i) = -\sum_{l\ne y_i} C(i,l), \\
&\exists h \in \mathcal{H} : \sum_{i=1}^{m} C(i,h(x_i)) \le -\frac{\gamma}{2} \sum_{i=1}^{m} \left\{ -C(i,y_i) + \sum_{l\ne y_i} C(i,l) \right\}.
\end{aligned} \tag{8}$$

Defining $\mathcal{C}^{\mathrm{MR}}$ to be the collection of cost matrices satisfying the constraints in (8), the above condition is the same as

$$\forall \mathbf{C} \in \mathcal{C}^{\mathrm{MR}}, \exists h \in \mathcal{H} : \mathbf{C} \bullet \left(\mathbf{1}_h - \mathbf{B}_\gamma^{\mathrm{MR}}\right) \le 0,$$

where $\mathbf{B}_\gamma^{\mathrm{MR}}(i,l) = [\![l = y_i]\!]\gamma/2$.

3.2 A Curious Equivalence

We show that the weak learning conditions of AdaBoost.MH and AdaBoost.M1 are identical in our framework. This is surprising because the original motivations behind these algorithms were completely different. AdaBoost.M1 is a direct extension of binary AdaBoost to the multiclass setting, whereas AdaBoost.MH is based on the one-against-all multiclass to binary reduction. This equivalence is a sort of degeneracy, and arises because the weak classifiers being used predict single labels per example. With multilabel weak classifiers, for which AdaBoost.MH was originally designed, the equivalence no longer holds.

The proofs in this and later sections will make use of the following minimax result, that is a weaker version of Corollary 37.3.2 of Rockafellar (1970).

Theorem 1 *(Minimax Theorem) Let C, D be non-empty closed convex subsets of $\mathbb{R}^m, \mathbb{R}^n$ respectively, and let K be a linear function on $C \times D$. If either C or D is bounded, then*

$$\min_{v \in D} \max_{u \in C} K(u,v) = \max_{u \in C} \min_{v \in D} K(u,v).$$

Lemma 2 *A weak classifier space $\mathcal{H}$ satisfies $(\mathcal{C}^{M1}, \mathbf{B}_\gamma^{M1})$ if and only if it satisfies $(\mathcal{C}^{MH}, \mathbf{B}_\gamma^{MH})$.*

Note that $\mathcal{C}^{\mathrm{M1}}$ and $\mathcal{C}^{\mathrm{MH}}$ depend implicitly on the training set. This lemma is valid for all training sets.

Proof We will refer to $(\mathcal{C}^{\mathrm{M1}}, \mathbf{B}_\gamma^{\mathrm{M1}})$ by M1 and $(\mathcal{C}^{\mathrm{MH}}, \mathbf{B}_\gamma^{\mathrm{MH}})$ by MH for brevity. The proof is in three steps.

Step (i): If $\mathcal{H}$ satisfies MH, then it also satisfies M1. This follows since any constraint (4) imposed by M1 on $\mathcal{H}$ can be reproduced by MH by plugging the following values of $d(i,l)$ in (6)

$$d(i,l) = \begin{cases} d(i) & \text{if } l = y_i \\ 0 & \text{if } l \neq y_i. \end{cases}$$

Step (ii): If $\mathcal{H}$ satisfies M1, then there is a convex combination $\mathbf{H}_{\boldsymbol{\lambda}^*}$ of the matrices $\mathbf{1}_h \in \mathcal{H}$, defined as

$$\mathbf{H}_{\boldsymbol{\lambda}^*} \triangleq \sum_{h \in \mathcal{H}} \lambda^*(h) \mathbf{1}_h,$$

such that

$$\forall i : \left(\mathbf{H}_{\boldsymbol{\lambda}^*} - \mathbf{B}_\gamma^{\mathrm{M1}}\right)(i,l) \begin{cases} \geq 0 & \text{if } l = y_i \\ \leq 0 & \text{if } l \neq y_i. \end{cases} \tag{9}$$

Indeed, Theorem 1 yields

$$\min_{\boldsymbol{\lambda} \in \Delta(\mathcal{H})} \max_{\mathbf{C} \in \mathcal{C}^{\mathrm{M1}}} \mathbf{C} \bullet \left(\mathbf{H}_{\boldsymbol{\lambda}} - \mathbf{B}_\gamma^{\mathrm{M1}}\right) = \max_{\mathbf{C} \in \mathcal{C}^{\mathrm{M1}}} \min_{h \in \mathcal{H}} \mathbf{C} \bullet \left(\mathbf{1}_h - \mathbf{B}_\gamma^{\mathrm{M1}}\right) \leq 0, \tag{10}$$

where the inequality is a restatement of our assumption that $\mathcal{H}$ satisfies M1. If $\boldsymbol{\lambda}^*$ is a minimizer of the minimax expression, then $\mathbf{H}_{\boldsymbol{\lambda}^*}$ must satisfy

$$\forall i : \mathbf{H}_{\boldsymbol{\lambda}^*}(i,l) \begin{cases} \geq \frac{1}{2} + \frac{\gamma}{2} & \text{if } l = y_i \\ \leq \frac{1}{2} - \frac{\gamma}{2} & \text{if } l \neq y_i, \end{cases} \tag{11}$$

or else some choice of $\mathbf{C} \in \mathcal{C}^{\mathrm{M1}}$ can cause $\mathbf{C} \bullet \left(\mathbf{H}_{\boldsymbol{\lambda}^*} - \mathbf{B}_\gamma^{\mathrm{M1}}\right)$ to exceed 0. In particular, if $\mathbf{H}_{\boldsymbol{\lambda}^*}(i_0, l) < 1/2 + \gamma/2$, then

$$\left(\mathbf{H}_{\boldsymbol{\lambda}^*} - \mathbf{B}_\gamma^{\mathrm{M1}}\right)(i_0, y_{i_0}) < \sum_{l \neq y_{i_0}} \left(\mathbf{H}_{\boldsymbol{\lambda}^*} - \mathbf{B}_\gamma^{\mathrm{M1}}\right)(i_0, l).$$

Now, if we choose $\mathbf{C} \in \mathcal{C}^{\mathrm{M1}}$ as

$$C(i,l) = \begin{cases} 0 & \text{if } i \neq i_0 \\ 1 & \text{if } i = i_0, l \neq y_{i_0} \\ -1 & \text{if } i = i_0, l = y_{i_0}, \end{cases}$$

then,

$$\mathbf{C} \bullet \left(\mathbf{H}_{\boldsymbol{\lambda}^*} - \mathbf{B}_\gamma^{\mathrm{M1}}\right) = -\left(\mathbf{H}_{\boldsymbol{\lambda}^*} - \mathbf{B}_\gamma^{\mathrm{M1}}\right)(i_0, y_{i_0}) + \sum_{l \neq y_{i_0}} \left(\mathbf{H}_{\boldsymbol{\lambda}^*} - \mathbf{B}_\gamma^{\mathrm{M1}}\right)(i_0, l) > 0,$$

contradicting the inequality in (10). Therefore (11) holds. Equation (9), and thus Step (ii), now follows by observing that $\mathbf{B}_\gamma^{\mathrm{MH}}$, by definition, satisfies

$$\forall i : \mathbf{B}_\gamma^{\mathrm{MH}}(i,l) = \begin{cases} \frac{1}{2} + \frac{\gamma}{2} & \text{if } l = y_i \\ \frac{1}{2} - \frac{\gamma}{2} & \text{if } l \neq y_i. \end{cases}$$

Step (iii) If there is some convex combination $\mathcal{H}_{\boldsymbol{\lambda}^*}$ satisfying (9), then $\mathcal{H}$ satisfies MH. Recall that $\mathbf{B}^{\mathrm{MH}}$ consists of entries that are non-positive on the correct labels and non-negative for incorrect labels. Therefore, (9) implies

$$0 \geq \max_{\mathbf{C} \in \mathcal{C}^{\mathrm{MH}}} \mathbf{C} \bullet \left(\mathbf{H}_{\boldsymbol{\lambda}^*} - \mathbf{B}_\gamma^{\mathrm{MH}}\right) \geq \min_{\boldsymbol{\lambda} \in \Delta(\mathcal{H})} \max_{\mathbf{C} \in \mathcal{C}^{\mathrm{MH}}} \mathbf{C} \bullet \left(\mathbf{H}_{\boldsymbol{\lambda}} - \mathbf{B}_\gamma^{\mathrm{MH}}\right).$$

On the other hand, using Theorem 1 we have

$$\min_{\boldsymbol{\lambda} \in \Delta(\mathcal{H})} \max_{\mathbf{C} \in \mathcal{C}^{\mathrm{MH}}} \mathbf{C} \bullet \left(\mathbf{H}_{\boldsymbol{\lambda}} - \mathbf{B}_\gamma^{\mathrm{MH}}\right) = \max_{\mathbf{C} \in \mathcal{C}^{\mathrm{MH}}} \min_{h \in \mathcal{H}} \mathbf{C} \bullet \left(\mathbf{1}_h - \mathbf{B}_\gamma^{\mathrm{MH}}\right).$$

Combining the two, we get

$$0 \geq \max_{\mathbf{C} \in \mathcal{C}^{\mathrm{MH}}} \min_{h \in \mathcal{H}} \mathbf{C} \bullet \left(\mathbf{1}_h - \mathbf{B}_\gamma^{\mathrm{MH}}\right),$$

which is the same as saying that $\mathcal{H}$ satisfies MH's condition.

Steps (ii) and (iii) together imply that if $\mathcal{H}$ satisfies M1, then it also satisfies MH. Along with Step (i), this concludes the proof. ■

4. Necessary and Sufficient Weak-learning Conditions

The binary weak-learning condition has an appealing form: for any distribution over the examples, the weak classifier needs to achieve error not greater than that of a random player who guesses the correct answer with probability $1/2 + \gamma/2$. Further, this is the weakest condition under which boosting is possible as follows from a game-theoretic perspective (Freund and Schapire, 1996b; Rätsch and Warmuth, 2005) . Multiclass weak-learning conditions with similar properties are missing in the literature. In this section we show how our framework captures such conditions.

4.1 Edge-over-random Conditions

In the multiclass setting, we model a random player as a baseline predictor $\mathbf{B} \in \mathbb{R}^{m \times k}$ whose rows are distributions over the labels, $\mathbf{B}(i) \in \Delta\{1, \ldots, k\}$. The prediction on example i is a sample from $\mathbf{B}(i)$. We only consider the space of *edge-over-random* baselines $\mathcal{B}_\gamma^{\text{eor}} \subseteq \mathbb{R}^{m \times k}$ who have a faint clue about the correct answer. More precisely, any baseline $\mathbf{B} \in \mathcal{B}_\gamma^{\text{eor}}$ in this space is γ more likely to predict the correct label than an incorrect one on every example i: $\forall l \neq 1, B(i,1) \geq B(i,l) + \gamma$, with equality holding for some l, that is:

$$B(i,1) = \max\left\{B(i,l) + \gamma : l \neq 1\right\}.$$

Notice that the edge-over-random baselines are different from the baselines used by earlier weak learning conditions discussed in the previous section.

When $k = 2$, the space $\mathcal{B}_\gamma^{\text{eor}}$ consists of the unique player $\mathbf{U}_\gamma^{\text{bin}}$, and the binary weak-learning condition is given by $(\mathcal{C}^{\text{bin}}, \mathbf{U}_\gamma^{\text{bin}})$. The new conditions generalize this to $k > 2$. In particular, define $\mathcal{C}^{\text{eor}}$ to be the multiclass extension of $\mathcal{C}^{\text{bin}}$: any cost-matrix in $\mathcal{C}^{\text{eor}}$ should put the least cost on the correct label, that is, the rows of the cost-matrices should come from the set $\left\{\mathbf{c} \in \mathbb{R}^k : \forall l, c(1) \leq c(l)\right\}$. Then, for every baseline $\mathbf{B} \in \mathcal{B}_\gamma^{\text{eor}}$, we introduce the condition $(\mathcal{C}^{\text{eor}}, \mathbf{B})$, which we call an *edge-over-random* weak-learning condition. Since $\mathbf{C} \bullet \mathbf{B}$ is the expected cost of the edge-over-random baseline $\mathbf{B}$ on matrix $\mathbf{C}$, the constraints (2) imposed by the new condition essentially require better than random performance.

Also recall that we have assumed that the true label y_i of example i in our training set is always 1. Nevertheless, we may occasionally continue to refer to the true labels as y_i.

We now present the central results of this section. The seemingly mild edge-over-random conditions guarantee boostability, meaning weak classifiers that satisfy any one such condition can be combined to form a highly accurate combined classifier.

Theorem 3 (Sufficiency) *If a weak classifier space $\mathcal{H}$ satisfies a weak-learning condition $(\mathcal{C}^{eor}, \mathbf{B})$, for some $\mathbf{B} \in \mathcal{B}_\gamma^{eor}$, then $\mathcal{H}$ is boostable.*

Proof The proof is in the spirit of the ones in Freund and Schapire (1996b). Applying Theorem 1 yields

$$0 \geq \max_{\mathbf{C} \in \mathcal{C}^{\text{eor}}} \min_{h \in \mathcal{H}} \mathbf{C} \bullet (\mathbf{1}_h - \mathbf{B}) = \min_{\boldsymbol{\lambda} \in \Delta(\mathcal{H})} \max_{\mathbf{C} \in \mathcal{C}^{\text{eor}}} \mathbf{C} \bullet (\mathbf{H}_{\boldsymbol{\lambda}} - \mathbf{B}),$$

where the first inequality follows from the definition (2) of the weak-learning condition. Let $\boldsymbol{\lambda}^*$ be a minimizer of the min-max expression. Unless the first entry of each row of $(\mathbf{H}_{\boldsymbol{\lambda}^*} - \mathbf{B})$ is the largest, the right hand side of the min-max expression can be made arbitrarily large by choosing $\mathbf{C} \in \mathcal{C}^{\text{eor}}$ appropriately. For example, if in some row i, the j_0^{th} element is strictly larger than the first element, by choosing

$$C(i,j) = \begin{cases} -1 & \text{if } j = 1 \\ 1 & \text{if } j = j_0 \\ 0 & \text{otherwise,} \end{cases}$$

we get a matrix in $\mathcal{C}^{\text{eor}}$ which causes $\mathbf{C} \bullet (\mathbf{H}_{\boldsymbol{\lambda}^*} - \mathbf{B})$ to be equal to $C(i, j_0) - C(i,1) > 0$, an impossibility by the first inequality.

Therefore, the convex combination of the weak classifiers, obtained by choosing each weak classifier with weight given by $\boldsymbol{\lambda}^*$, perfectly classifies the training data, in fact with a margin γ. ■

On the other hand, the family of such conditions, taken as a whole, is necessary for boostability in the sense that every eligible space of weak classifiers satisfies some edge-over-random condition.

Theorem 4 (Relaxed necessity) *For every boostable weak classifier space $\mathcal{H}$, there exists a $\gamma > 0$ and $\mathbf{B} \in \mathcal{B}_\gamma^{eor}$ such that $\mathcal{H}$ satisfies the weak-learning condition $(\mathcal{C}^{eor}, \mathbf{B})$.*

Proof The proof shows existence through non-constructive averaging arguments. We will reuse notation from the proof of Theorem 3 above. $\mathcal{H}$ is boostable implies there exists some distribution $\boldsymbol{\lambda}^* \in \Delta(\mathcal{H})$ such that

$$\forall j \neq 1, i : \mathbf{H}_{\boldsymbol{\lambda}^*}(i,1) - \mathbf{H}_{\boldsymbol{\lambda}^*}(i,j) > 0.$$

Let $\gamma > 0$ be the minimum of the above expression over all possible (i,j), and let $\mathbf{B} = \mathbf{H}_{\boldsymbol{\lambda}^*}$. Then $\mathbf{B} \in \mathcal{B}_\gamma^{\text{eor}}$, and

$$\max_{\mathbf{C}\in\mathcal{C}^{\text{eor}}} \min_{h\in\mathcal{H}} \mathbf{C} \bullet (\mathbf{1}_h - \mathbf{B}) \leq \min_{\boldsymbol{\lambda}\in\Delta(\mathcal{H})} \max_{\mathbf{C}\in\mathcal{C}^{\text{eor}}} \mathbf{C} \bullet (\mathbf{H}_{\boldsymbol{\lambda}} - \mathbf{B}) \leq \max_{\mathbf{C}\in\mathcal{C}^{\text{eor}}} \mathbf{C} \bullet (\mathbf{H}_{\boldsymbol{\lambda}^*} - \mathbf{B}) = 0,$$

where the equality follows since by definition $\mathbf{H}_{\boldsymbol{\lambda}^*} - \mathbf{B} = \mathbf{0}$. The max-min expression is at most zero is another way of saying that $\mathcal{H}$ satisfies the weak-learning condition $(\mathcal{C}^{\text{eor}}, \mathbf{B})$ as in (2). ■

Theorem 4 states that any boostable weak classifier space will satisfy some condition in our family, but it does not help us choose the right condition. Experiments in Section 10 suggest $\left(\mathcal{C}^{\text{eor}}, \mathbf{U}_\gamma\right)$ is effective with very simple weak-learners compared to popular boosting algorithms. (Recall $\mathbf{U}_\gamma \in \mathcal{B}_\gamma^{\text{eor}}$ is the edge-over-random baseline closest to uniform; it has weight $(1-\gamma)/k$ on incorrect labels and $(1-\gamma)/k + \gamma$ on the correct label.) However, there are theoretical examples showing each condition in our family is too strong.

Theorem 5 *For any $\mathbf{B} \in \mathcal{B}_\gamma^{eor}$, there exists a boostable space $\mathcal{H}$ that fails to satisfy the condition $(\mathcal{C}^{eor}, \mathbf{B})$.*

Proof We provide, for any $\gamma > 0$ and edge-over-random baseline $\mathbf{B} \in \mathcal{B}_\gamma^{\text{eor}}$, a data set and weak classifier space that is boostable but fails to satisfy the condition $(\mathcal{C}^{\text{eor}}, \mathbf{B})$.

Pick $\gamma' = \gamma/k$ and set $m > 1/\gamma'$ so that $\lfloor m(1/2+\gamma') \rfloor > m/2$. Our data set will have m labeled examples $\{(0, y_0), \ldots, (m-1, y_{m-1})\}$, and m weak classifiers. We want the following symmetries in our weak classifiers:

- Each weak classifier correctly classifies $\lfloor m(1/2+\gamma') \rfloor$ examples and misclassifies the rest.
- On each example, $\lfloor m(1/2+\gamma') \rfloor$ weak classifiers predict correctly.

Note the second property implies boostability, since the uniform convex combination of all the weak classifiers is a perfect predictor.

The two properties can be satisfied by the following design. A window is a contiguous sequence of examples that may wrap around; for example

$$\{i, (i+1) \mod m, \ldots, (i+k) \mod m\}$$

is a window containing k elements, which may wrap around if $i + k \geq m$. For each window of length $\lfloor m(1/2+\gamma') \rfloor$ create a hypothesis that correctly classifies within the window, and misclassifies outside. This weak-hypothesis space has size m, and has the required properties.

We still have flexibility as to how the misclassifications occur, and which cost-matrix to use, which brings us to the next two choices:

- Whenever a hypothesis misclassifies on example i, it predicts label

$$\hat{y}_i \stackrel{\triangle}{=} \operatorname{argmin}\{B(i,l) : l \neq y_i\}. \tag{12}$$

- A cost-matrix is chosen so that the cost of predicting $\hat{y}_i$ on example i is 1, but for any other prediction the cost is zero. Observe this cost-matrix belongs to $\mathcal{C}^{\text{eor}}$.

Therefore, every time a weak classifier predicts incorrectly, it also suffers cost 1. Since each weak classifier predicts correctly only within a window of length $\lfloor m(1/2+\gamma')\rfloor$, it suffers cost $\lceil m(1/2-\gamma')\rceil$. On the other hand, by the choice of $\hat{y}_i$ in (12), and by our assumption that $y_i = 1$, we have

$$\begin{aligned} B(i,\hat{y}_i) &= \min\{B(i,1)-\gamma, B(i,2),\ldots,B(i,k)\} \\ &\leq \frac{1}{k}\left(B(i,1)-\gamma+B(i,2)+B(i,3)+\ldots+B(i,k)\right) \\ &= 1/k-\gamma/k. \end{aligned}$$

So the cost of $\mathbf{B}$ on the chosen cost-matrix is at most $m(1/k-\gamma/k)$, which is less than the cost $\lceil m(1/2-\gamma')\rceil \geq m(1/2-\gamma/k)$ of any weak classifier whenever the number of labels k is more than two. Hence our boostable space of weak classifiers fails to satisfy $(\mathcal{C}^{\text{eor}}, \mathbf{B})$. ■

Theorems 4 and 5 can be interpreted as follows. While a boostable space will satisfy *some* edge-over-random condition, without further information about the data set it is not possible to know *which* particular condition will be satisfied. The kind of prior knowledge required to make this guess correctly is provided by Theorem 3: the appropriate weak learning condition is determined by the distribution of votes on the labels for each example that a target weak classifier combination might be able to get. Even with domain expertise, such knowledge may or may not be obtainable in practice before running boosting. We therefore need conditions that assume less.

4.2 The Minimal Weak Learning Condition

A perhaps extreme way of weakening the condition is by requiring the performance on a cost matrix to be competitive not with a *fixed* baseline $\mathbf{B} \in \mathcal{B}_\gamma^{\text{eor}}$, but with the *worst* of them:

$$\forall \mathbf{C} \in \mathcal{C}^{\text{eor}}, \exists h \in \mathcal{H} : \mathbf{C} \bullet \mathbf{1}_h \leq \max_{\mathbf{B} \in \mathcal{B}_\gamma^{\text{eor}}} \mathbf{C} \bullet \mathbf{B}. \tag{13}$$

Condition (13) states that during the course of the same boosting game, Weak-Learner may choose to beat *any* edge-over-random baseline $\mathbf{B} \in \mathcal{B}_\gamma^{\text{eor}}$, possibly a different one for every round and every cost-matrix. This may superficially seem much too weak. On the contrary, this condition turns out to be equivalent to boostability. In other words, according to our criterion, it is neither too weak nor too strong as a weak-learning condition. However, unlike the edge-over-random conditions, it also turns out to be more difficult to work with algorithmically.

Furthermore, this condition can be shown to be equivalent to the one used by AdaBoost.MR (Schapire and Singer, 1999; Freund and Schapire, 1996a). This is perhaps remarkable since the latter is based on the apparently completely unrelated all-pairs multiclass to binary reduction. In Section 3 we saw that the MR condition is given by $(\mathcal{C}^{\text{MR}}, \mathbf{B}_\gamma^{\text{MR}})$, where $\mathcal{C}^{\text{MR}}$ consists of cost-matrices that put non-negative costs on incorrect labels and whose rows sum up to zero, while $\mathbf{B}_\gamma^{\text{MR}} \in \mathbb{R}^{m\times k}$ is the matrix that has γ on the first column and $-\gamma$ on all other columns. Further, the MR condition, and hence (13), can be shown to be neither too weak nor too strong.

Theorem 6 (MR) *A weak classifier space $\mathcal{H}$ satisfies AdaBoost.MR's weak-learning condition* $(\mathcal{C}^{MR}, \mathbf{B}_\gamma^{MR})$ *if and only if it satisfies* (13)*. Moreover, this condition is equivalent to being boostable.*

Proof We will show the following three conditions are equivalent:

(A) $\mathcal{H}$ is boostable

(B) $\exists\gamma > 0$ such that $\forall \mathbf{C} \in \mathcal{C}^{\text{eor}}, \exists h \in \mathcal{H} : \mathbf{C} \bullet \mathbf{1}_h \leq \max_{\mathbf{B} \in \mathcal{B}_\gamma^{\text{eor}}} \mathbf{C} \bullet \mathbf{B}$

(C) $\exists\gamma > 0$ such that $\forall \mathbf{C} \in \mathcal{C}^{\text{MR}}, \exists h \in \mathcal{H} : \mathbf{C} \bullet \mathbf{1}_h \leq \mathbf{C} \bullet \mathbf{B}^{\text{MR}}$.

We will show (A) implies (B), (B) implies (C), and (C) implies (A) to achieve the above.

(A) implies (B): Immediate from Theorem 4.

(B) implies (C): Suppose (B) is satisfied with 2γ. We will show that this implies $\mathcal{H}$ satisfies $(\mathcal{C}^{\text{MR}}, \mathbf{B}_\gamma^{\text{MR}})$. Notice $\mathcal{C}^{\text{MR}} \subset \mathcal{C}^{\text{eor}}$. Therefore it suffices to show that

$$\forall \mathbf{C} \in \mathcal{C}^{\text{MR}}, \mathbf{B} \in \mathcal{B}_{2\gamma}^{\text{eor}} : \mathbf{C} \bullet \left(\mathbf{B} - \mathbf{B}_\gamma^{\text{MR}}\right) \leq 0.$$

Notice that $\mathbf{B} \in \mathcal{B}_{2\gamma}^{\text{eor}}$ implies $\mathbf{B}' = \mathbf{B} - \mathbf{B}_\gamma^{\text{MR}}$ is a matrix whose largest entry in each row is in the first column of that row. Then, for any $\mathbf{C} \in \mathcal{C}^{\text{MR}}$, $\mathbf{C} \bullet \mathbf{B}'$ can be written as

$$\mathbf{C} \bullet \mathbf{B}' = \sum_{i=1}^{m} \sum_{j=2}^{k} C(i,j) \left(B'(i,j) - B'(i,1)\right).$$

Since $C(i,j) \geq 0$ for $j > 1$, and $B'(i,j) - B'(i,1) \leq 0$, we have our result.

(C) implies (A): Applying Theorem 1

$$\begin{aligned} 0 &\geq \max_{\mathbf{C} \in \mathcal{C}^{\text{MR}}} \min_{h \in \mathcal{H}} \mathbf{C} \bullet \left(\mathbf{1}_h - \mathbf{B}_\gamma^{\text{MR}}\right) \\ &\geq \max_{\mathbf{C} \in \mathcal{C}^{\text{MR}}} \min_{\boldsymbol{\lambda} \in \Delta(\mathcal{H})} \mathbf{C} \bullet \left(\mathbf{H}_{\boldsymbol{\lambda}} - \mathbf{B}_\gamma^{\text{MR}}\right) \\ &= \min_{\boldsymbol{\lambda} \in \Delta(\mathcal{H})} \max_{\mathbf{C} \in \mathcal{C}^{\text{MR}}} \mathbf{C} \bullet \left(\mathbf{H}_{\boldsymbol{\lambda}} - \mathbf{B}_\gamma^{\text{MR}}\right). \end{aligned}$$

For any i_0 and $l_0 \neq 1$, the following cost-matrix $\mathbf{C}$ satisfies $\mathbf{C} \in \mathcal{C}^{\text{MR}}$,

$$C(i,l) = \begin{cases} 0 & \text{if } i \neq i_0 \text{ or } l \notin \{1, l_0\} \\ 1 & \text{if } i = i_0, l = l_0 \\ -1 & \text{if } i = i_0, l = 1. \end{cases}$$

Let $\boldsymbol{\lambda}$ belong to the argmin of the min max expression. Then $\mathbf{C} \bullet \left(\mathbf{H}_{\boldsymbol{\lambda}} - \mathbf{B}_\gamma^{\text{MR}}\right) \leq 0$ implies $\mathbf{H}_{\boldsymbol{\lambda}}(i_0, 1) - \mathbf{H}_{\boldsymbol{\lambda}}(i_0, l_0) \geq 2\gamma$. Since this is true for all i_0 and $l_0 \neq 1$, we conclude that the $(\mathcal{C}^{\text{MR}}, \mathbf{B}_\gamma^{\text{MR}})$ condition implies boostability.

This concludes the proof of equivalence. ■

Next, we illustrate the strengths of our minimal weak-learning condition through concrete comparisons with previous algorithms.

	h_1	h_2
a	1	2
b	1	2

Figure 1: A weak classifier space which satisfies SAMME's weak learning condition but is not boostable.

4.2.1 COMPARISON WITH SAMME

The SAMME algorithm of Zhu et al. (2009) requires the weak classifiers to achieve less error than uniform random guessing for multiple labels; in our language, their weak-learning condition is $(\mathcal{C}^{\text{SAM}}, \mathbf{U}_\gamma)$, as shown in Section 3, where $\mathcal{C}^{\text{SAM}}$ consists of cost matrices whose rows are of the form $(0,t,t,\ldots)$ for some non-negative t. As is well-known, this condition is not sufficient for boosting to be possible. In particular, consider the data set $\{(a,1),(b,2)\}$ with $k=3, m=2$, and a weak classifier space consisting of h_1, h_2 which always predict $1,2$, respectively (Figure 1). Since neither classifier distinguishes between a,b we cannot achieve perfect accuracy by combining them in any way. Yet, due to the constraints on the cost-matrix, one of h_1, h_2 will always manage non-positive cost while random always suffers positive cost. On the other hand our weak-learning condition allows the Booster to choose far richer cost matrices. In particular, when the cost matrix $\mathbf{C} \in \mathcal{C}^{\text{eor}}$ is given by

$$\begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline a & -1 & +1 & 0 \\ b & +1 & -1 & 0, \end{array}$$

both classifiers in the above example suffer more loss than the random player $\mathbf{U}_\gamma$, and fail to satisfy our condition.

4.2.2 COMPARISON WITH ADABOOST.MH

AdaBoost.MH (Schapire and Singer, 1999) was designed for use with weak hypotheses that on each example return a prediction for every label. When used in our framework, where the weak classifiers return only a single multiclass prediction per example, the implicit demands made by AdaBoost.MH on the weak classifier space turn out to be too strong. We cannot use Theorem 5 to demonstrate this, since it applies to only fixed edge-over-random conditions. Instead, we construct a classifier space that satisfies the condition $(\mathcal{C}^{\text{eor}}, \mathbf{U}_\gamma)$ in our family, but cannot satisfy AdaBoost.MH's weak-learning condition. Note that this does not imply that the conditions are too strong when used with more powerful weak classifiers that return multilabel multiclass predictions.

Consider a space $\mathcal{H}$ that has, for every $(1/k+\gamma)m$ element subset of the examples, a classifier that predicts correctly on exactly those elements. The expected loss of a randomly chosen classifier from this space is the same as that of the random player $\mathbf{U}_\gamma$. Hence $\mathcal{H}$ satisfies this weak-learning condition. On the other hand, it was shown in Section 3 that AdaBoost.MH's weak-learning condition is the pair $(\mathcal{C}^{\text{MH}}, \mathbf{B}_\gamma^{\text{MH}})$, where $\mathcal{C}^{\text{MH}}$ consists of cost matrices with non-negative entries on incorrect labels and non-positive entries on real labels, and where each row of the matrix $\mathbf{B}_\gamma^{\text{MH}}$ is the vector $(1/2+\gamma/2, 1/2-\gamma/2, \ldots, 1/2-\gamma/2)$. A quick calculation shows that for any $h \in \mathcal{H}$, and

$\mathbf{C} \in \mathcal{C}^{\mathrm{MH}}$ with -1 in the first column and zeroes elsewhere, $\mathbf{C} \bullet \left(\mathbf{1}_h - \mathbf{B}^{\mathrm{MH}}_\gamma\right) = 1/2 - 1/k$. This is positive when $k > 2$, so that $\mathcal{H}$ fails to satisfy AdaBoost.MH's condition.

We have seen how our framework allows us to capture the strengths and weaknesses of old conditions, describe a whole new family of conditions and also identify the condition making minimal assumptions. In the next few sections, we show how to design boosting algorithms that employ these new conditions and enjoy strong theoretical guarantees.

5. Algorithms

In this section we devise algorithms by analyzing the boosting games that employ weak-learning conditions in our framework. We compute the optimum Booster strategy against a completely adversarial Weak-Learner, which here is permitted to choose weak classifiers without restriction, that is, the entire space $\mathcal{H}^{\mathrm{all}}$ of all possible functions mapping examples to labels. By modeling Weak-Learner adversarially, we make absolutely no assumptions on the algorithm it might use. Hence, error guarantees enjoyed in this situation will be universally applicable. Our algorithms are derived from the very general drifting games framework (Schapire, 2001) for solving boosting games, which in turn was inspired by Freund's Boost-by-majority algorithm (Freund, 1995), which we review next.

5.1 The OS Algorithm

Fix the number of rounds T and a weak-learning condition $(\mathcal{C}, \mathbf{B})$. We will only consider conditions that are not *vacuous*, that is, at least some classifier space satisfies the condition, or equivalently, the space $\mathcal{H}^{\mathrm{all}}$ satisfies $(\mathcal{C}, \mathbf{B})$. Additionally, we assume the constraints placed by $\mathcal{C}$ are on individual rows. In other words, there is some subset $\mathcal{C}_0 \subseteq \mathbb{R}^k$ of all possible rows, such that a cost matrix $\mathbf{C}$ belongs to the collection $\mathcal{C}$ if and only if each of its rows belongs to this subset:

$$\mathbf{C} \in \mathcal{C} \iff \forall i : \mathbf{C}(i) \in \mathcal{C}_0. \tag{14}$$

Further, we assume $\mathcal{C}_0$ forms a convex cone, that is, $\mathbf{c}, \mathbf{c}' \in \mathcal{C}_0$ implies $t\mathbf{c} + t'\mathbf{c}' \in \mathcal{C}_0$ for any non-negative t, t'. This also implies that $\mathcal{C}$ is a convex cone. This is a very natural restriction, and is satisfied by the space $\mathbf{C}$ used by the weak learning conditions of AdaBoost.MH, AdaBoost.M1, AdaBoost.MR, SAMME as well as every edge-over-random condition.[1] For simplicity of presentation we fix the weights $\alpha_t = 1$ in each round. With $\mathbf{f}_T$ defined as in (1), whether the final hypotheses output by Booster makes a prediction error on an example i is decided by whether an incorrect label received the maximum number of votes, $f_T(i,1) \leq \max_{l=2}^{k} f_T(i,l)$. Therefore, the optimum Booster payoff can be written as

$$\min_{\mathbf{C}_1 \in \mathcal{C}} \max_{\substack{h_1 \in \mathcal{H}^{\mathrm{all}}: \\ \mathbf{C}_1 \bullet \left(\mathbf{1}_{h_1} - \mathbf{B}\right) \leq \mathbf{0}}} \ldots \min_{\mathbf{C}_T \in \mathcal{C}} \max_{\substack{h_T \in \mathcal{H}^{\mathrm{all}}: \\ \mathbf{C}_T \bullet \left(\mathbf{1}_{h_T} - \mathbf{B}\right) \leq \mathbf{0}}} \frac{1}{m} \sum_{i=1}^{m} L^{\mathrm{err}}(f_T(x_i,1), \ldots, f_T(x_i,k)). \tag{15}$$

where the function $L^{\mathrm{err}} : \mathbb{R}^k \to \mathbb{R}$ encodes 0-1 error

$$L^{\mathrm{err}}(\mathbf{s}) = \mathbf{1}\left[s(1) \leq \max_{l>1} s(l)\right]. \tag{16}$$

[1] All our results hold under the weaker restriction on the space $\mathcal{C}$, where the set of possible cost vectors $\mathcal{C}_0$ for a row i could depend on i. For simplicity of exposition, we stick to the more restrictive assumption that $\mathcal{C}_0$ is common across all rows.

In general, we will also consider other loss functions $L : \mathbb{R}^k \to \mathbb{R}$ such as exponential loss, hinge loss, etc. that upper-bound error and are *proper*: that is, $L(\mathbf{s})$ is increasing in the weight of the correct label $s(1)$, and decreasing in the weights of the incorrect labels $s(l), l \neq 1$.

Directly analyzing the optimal payoff is hard. However, Schapire (2001) observed that the payoffs can be very well approximated by certain potential functions. Indeed, for any $\mathbf{b} \in \mathbb{R}^k$ define the *potential function* $\phi_t^{\mathbf{b}} : \mathbb{R}^k \to \mathbb{R}$ by the following recurrence:

$$\begin{array}{rcl} \phi_0^{\mathbf{b}}(\mathbf{s}) & = & L(\mathbf{s}) \\ \phi_t^{\mathbf{b}}(\mathbf{s}) & = & \min\limits_{\mathbf{c}\in\mathcal{C}_0} \begin{array}{cc} \max\limits_{\mathbf{p}\in\Delta\{1,\ldots,k\}} & \mathbb{E}_{l\sim\mathbf{p}}\left[\phi_{t-1}^{\mathbf{b}}\left(\mathbf{s}+\mathbf{e}_l\right)\right] \\ \text{s.t.} & \mathbb{E}_{l\sim\mathbf{p}}\left[c(l)\right] \leq \langle \mathbf{b}, \mathbf{c}\rangle , \end{array} \end{array} \tag{17}$$

where $l \sim \mathbf{p}$ denotes that label l is sampled from the distribution $\mathbf{p}$, and $\mathbf{e}_l \in \mathbb{R}^k$ is the unit-vector whose lth coordinate is 1 and the remaining coordinates zero. Notice the recurrence uses the collection of rows $\mathcal{C}_0$ instead of the collection of cost matrices $\mathcal{C}$. When there are $T-t$ rounds remaining (that is, after t rounds of boosting), these potential functions compute an estimate $\phi_{T-t}^{\mathbf{b}}(\mathbf{s}_t)$ of whether an example x will be misclassified, based on its current state $\mathbf{s}_t$ consisting of counts of votes received so far on various classes:

$$s_t(l) = \sum_{t'=1}^{t-1} \mathbf{1}\left[h_{t'}(x) = l\right]. \tag{18}$$

Notice this definition of state assumes that $\alpha_t = 1$ in each round. Sometimes, we will choose the weights differently. In such cases, a more appropriate definition is the weighted state $\mathbf{f}_t \in \mathbb{R}^k$, tracking the weighted counts of votes received so far:

$$f_t(l) = \sum_{t'=1}^{t-1} \alpha_{t'} \mathbf{1}\left[h_{t'}(x) = l\right]. \tag{19}$$

However, unless otherwise noted, we will assume $\alpha_t = 1$, and so the definition in (18) will suffice.

The recurrence in (17) requires the max player's response $\mathbf{p}$ to satisfy the constraint that the expected cost under the distribution $\mathbf{p}$ is at most the inner-product $\langle \mathbf{c}, \mathbf{b}\rangle$. If there is no distribution satisfying this requirement, then the value of the max expression is $-\infty$. The existence of a valid distribution depends on both $\mathbf{b}$ and $\mathbf{c}$ and is captured by the following:

$$\exists \mathbf{p} \in \Delta\{1,\ldots,k\} : \mathbb{E}_{l\sim\mathbf{p}}\left[c(l)\right] \leq \langle \mathbf{c}, \mathbf{b}\rangle \iff \min_l c(l) \leq \langle \mathbf{b}, \mathbf{c}\rangle . \tag{20}$$

In this paper, the vector $\mathbf{b}$ will always correspond to some row $\mathbf{B}(i)$ of the baseline used in the weak learning condition. In such a situation, the next lemma shows that a distribution satisfying the required constraints will always exist.

Lemma 7 *If $\mathcal{C}_0$ is a cone and* (14) *holds, then for any row $\mathbf{b} = \mathbf{B}(i)$ of the baseline and any cost vector $\mathbf{c} \in \mathcal{C}_0$,* (20) *holds unless the condition $(\mathcal{C}, \mathbf{B})$ is vacuous.*

Proof We show that if (20) does not hold, then the condition is vacuous. Assume that for row $\mathbf{b} = \mathbf{B}(i_0)$ of the baseline, and some choice of cost vector $\mathbf{c} \in \mathcal{C}_0$, (20) does not hold. We pick a cost-matrix $\mathbf{C} \in \mathcal{C}$, such that no weak classifier h can satisfy the requirement (2), implying the condition must be vacuous. The i_0^{th} row of the cost matrix is $\mathbf{c}$, and the remaining rows are $\mathbf{0}$. Since $\mathcal{C}_0$ is a cone, $\mathbf{0} \in \mathcal{C}_0$ and hence the cost matrix lies in $\mathcal{C}$. With this choice for $\mathbf{C}$, the condition (2) becomes

$$c(h(x_i)) = C(i, h(x_i)) \le \langle \mathbf{C}(i), \mathbf{B}(i) \rangle = \langle \mathbf{c}, \mathbf{b} \rangle < \min_l c(l),$$

where the last inequality holds since, by assumption, (20) is not true for this choice of $\mathbf{c}, \mathbf{b}$. The previous equation is an impossibility, and hence no such weak classifier h exists, showing the condition is vacuous. ■

Lemma 7 shows that the expression in (17) is well defined, and takes on finite values. We next record an alternate dual form for the same recurrence which will be useful later.

Lemma 8 *The recurrence in* (17) *is equivalent to*

$$\phi_t^{\mathbf{b}}(\mathbf{s}) = \min_{\mathbf{c} \in \mathcal{C}_0} \max_{l=1}^{k} \left\{ \phi_{t-1}^{\mathbf{b}}(\mathbf{s} + \mathbf{e}_l) - (c(l) - \langle \mathbf{c}, \mathbf{b} \rangle) \right\}. \tag{21}$$

Proof Using Lagrangean multipliers, we may convert (17) to an unconstrained expression as follows:

$$\phi_t^{\mathbf{b}}(\mathbf{s}) = \min_{\mathbf{c} \in \mathcal{C}_0} \max_{\mathbf{p} \in \Delta\{1,\ldots,k\}} \min_{\lambda \ge 0} \left\{ \mathbb{E}_{l \sim \mathbf{p}} \left[\phi_{t-1}^{\mathbf{b}}(\mathbf{s} + \mathbf{e}_l) \right] - \lambda \left(\mathbb{E}_{l \sim \mathbf{p}} [c(l)] - \langle \mathbf{c}, \mathbf{b} \rangle \right) \right\}.$$

Applying Theorem 1 to the inner min-max expression we get

$$\phi_t^{\mathbf{b}}(\mathbf{s}) = \min_{\mathbf{c} \in \mathcal{C}_0} \min_{\lambda \ge 0} \max_{\mathbf{p} \in \Delta\{1,\ldots,k\}} \left\{ \mathbb{E}_{l \sim \mathbf{p}} \left[\phi_{t-1}^{\mathbf{b}}(\mathbf{s} + \mathbf{e}_l) \right] - \left(\mathbb{E}_{l \sim \mathbf{p}} [\lambda c(l)] - \langle \lambda \mathbf{c}, \mathbf{b} \rangle \right) \right\}.$$

Since $\mathcal{C}_0$ is a cone, $\mathbf{c} \in \mathcal{C}_0$ implies $\lambda \mathbf{c} \in \mathcal{C}_0$. Therefore we may absorb the Lagrange multiplier into the cost vector:

$$\phi_t^{\mathbf{b}}(\mathbf{s}) = \min_{\mathbf{c} \in \mathcal{C}_0} \max_{\mathbf{p} \in \Delta\{1,\ldots,k\}} \mathbb{E}_{l \sim \mathbf{p}} \left[\phi_{t-1}^{\mathbf{b}}(\mathbf{s} + \mathbf{e}_l) - (c(l) - \langle \mathbf{c}, \mathbf{b} \rangle) \right].$$

For a fixed choice of $\mathbf{c}$, the expectation is maximized when the distribution $\mathbf{p}$ is concentrated on a single label that maximizes the inner expression, which completes our proof. ■

The dual form of the recurrence is useful for optimally choosing the cost matrix in each round. When the weak learning condition being used is $(\mathcal{C}, \mathbf{B})$, Schapire (2001) proposed a Booster strategy, called the OS strategy, which always chooses the weight $\alpha_t = 1$, and uses the potential functions to construct a cost matrix $\mathbf{C}_t$ as follows. Each row $\mathbf{C}_t(i)$ of the matrix achieves the minimum of the right hand side of (21) with $\mathbf{b}$ replaced by $\mathbf{B}(i)$, t replaced by $T - t$, and $\mathbf{s}$ replaced by current state $\mathbf{s}_t(i)$:

$$\mathbf{C}_t(i) = \operatorname*{argmin}_{\mathbf{c} \in \mathcal{C}_0} \max_{l=1}^{k} \left\{ \phi_{T-t-1}^{\mathbf{B}(i)}(\mathbf{s} + \mathbf{e}_l) - (c(l) - \langle \mathbf{c}, \mathbf{B}(i) \rangle) \right\}. \tag{22}$$

The following theorem, proved in the appendix, provides a guarantee for the loss suffered by the OS algorithm, and also shows that it is the game-theoretically optimum strategy when the number of examples is large. Similar results have been proved by Schapire (2001), but our theorem holds much more generally, and also achieves tighter lower bounds.

Theorem 9 (Extension of results in Schapire (2001)) *Suppose the weak-learning condition is not vacuous and is given by* $(\mathcal{C},\mathbf{B})$, *where* $\mathcal{C}$ *is such that, for some convex cone* $\mathcal{C}_0 \subseteq \mathbb{R}^k$, *the condition* (14) *holds. Let the potential functions* $\phi_t^{\mathbf{b}}$ *be defined as in* (17), *and assume the Booster employs the OS algorithm, choosing* $\alpha_t = 1$ *and* $\mathbf{C}_t$ *as in* (22) *in each round t. Then the average potential of the states,*

$$\frac{1}{m}\sum_{i=1}^{m}\phi_{T-t}^{\mathbf{B}(i)}\left(\mathbf{s}_t(i)\right),$$

never increases in any round. In particular, the loss suffered after T rounds of play is at most

$$\frac{1}{m}\sum_{i=1}^{m}\phi_{T}^{\mathbf{B}(i)}(\mathbf{0}). \tag{23}$$

Further, under certain conditions, this bound is nearly tight. In particular, assume the loss function does not vary too much but satisfies

$$\sup_{\mathbf{s},\mathbf{s}'\in\mathcal{S}_T}\left|L(\mathbf{s})-L(\mathbf{s}')\right| \le \varnothing(L,T), \tag{24}$$

where $\mathcal{S}_T$*, a subset of* $\left\{\mathbf{s}\in\mathbb{R}^k : \|\mathbf{s}\|_\infty \le T\right\}$*, is the set of all states reachable in T iterations, and* $\varnothing(L,T)$ *is an upper bound on the discrepancy of losses between any two reachable states when the loss function is L and the total number of iterations is T. Then, for any* $\varepsilon > 0$*, when the number of examples m is sufficiently large,*

$$m \ge \frac{T\varnothing(L,T)}{\varepsilon}, \tag{25}$$

no Booster strategy can guarantee to achieve in T rounds a loss that is ε *less than the bound* (23).

In order to implement the nearly optimal OS strategy, we need to solve (22). This is computationally only as hard as evaluating the potentials, which in turn reduces to computing the recurrences in (17). In the next few sections, we study how to do this when using various losses and weak learning conditions.

6. Solving for Any Fixed Edge-over-random Condition

In this section we show how to implement the OS strategy when the weak learning condition is any fixed edge-over-random condition: $(\mathcal{C},\mathbf{B})$ for some $\mathbf{B}\in\mathcal{B}_\gamma^{\text{eor}}$. By our previous discussions, this is equivalent to computing the potential $\phi_t^{\mathbf{b}}$ by solving the recurrence in (17), where the vector $\mathbf{b}$ corresponds to some row of the baseline $\mathbf{B}$. Let $\Delta_\gamma^k \subseteq \Delta\{1,\ldots,k\}$ denote the set of all edge-over-random distributions on $\{1,\ldots,k\}$ with γ more weight on the first coordinate:

$$\Delta_\gamma^k = \left\{\mathbf{b}\in\Delta\{1,\ldots,k\} : b(1)-\gamma = \max\left\{b(2),\ldots,b(k)\right\}\right\}. \tag{26}$$

Note, that $\mathcal{B}_\gamma^{\text{eor}}$ consists of all matrices whose rows belong to the set Δ_γ^k. Therefore we are interested in computing $\phi^{\mathbf{b}}$, where $\mathbf{b}$ is an arbitrary edge-over-random distribution: $\mathbf{b}\in\Delta_\gamma^k$. We begin by simplifying the recurrence (17) satisfied by such potentials, and show how to compute the optimal cost matrix in terms of the potentials.

Lemma 10 *Assume L is proper, and $\mathbf{b} \in \Delta_\gamma^k$ is an edge-over-random distribution. Then the recurrence* (17) *may be simplified as*

$$\phi_t^{\mathbf{b}}(\mathbf{s}) \quad = \quad \mathbb{E}_{l \sim \mathbf{b}}\left[\phi_{t-1}(\mathbf{s}+\mathbf{e}_l)\right]. \tag{27}$$

Further, if the cost matrix $\mathbf{C}_t$ is chosen as follows

$$C_t(i,l) = \phi_{T-t-1}^{\mathbf{b}}(\mathbf{s}_t(i)+\mathbf{e}_l), \tag{28}$$

then $\mathbf{C}_t$ satisfies the condition in (22)*, and hence is the optimal choice.*

Proof Let $\mathcal{C}_0^{\text{eor}} \subseteq \mathbb{R}^k$ denote all vectors $\mathbf{c}$ satisfying $\forall l : c(1) \leq c(l)$. Then, we have

$$\begin{aligned}
\phi_t^{\mathbf{b}}(\mathbf{s}) &= \min_{\mathbf{c} \in \mathcal{C}_0^{\text{eor}}} \begin{array}{cc} \max\limits_{\mathbf{p} \in \Delta\{1,\ldots,k\}} & \mathbb{E}_{l \sim \mathbf{p}}\left[\phi_{t-1}(\mathbf{s}+\mathbf{e}_l)\right] \\ \text{s.t.} & \mathbb{E}_{l\sim\mathbf{p}}[c(l)] \leq \mathbb{E}_{l\sim\mathbf{b}}[c(l)], \end{array} \quad \text{(by (17))} \\
&= \min_{\mathbf{c} \in \mathcal{C}_0^{\text{eor}}} \max_{\mathbf{p}\in\Delta} \min_{\lambda \geq 0} \left\{\mathbb{E}_{l\sim\mathbf{p}}\left[\phi_{t-1}^{\mathbf{b}}(\mathbf{s}+\mathbf{e}_l)\right] + \lambda\left(\mathbb{E}_{l\sim\mathbf{b}}[c(l)] - \mathbb{E}_{l\sim\mathbf{p}}[c(l)]\right)\right\} \text{ (Lagrangean)} \\
&= \min_{\mathbf{c} \in \mathcal{C}_0^{\text{eor}}} \min_{\lambda\geq 0} \max_{\mathbf{p}\in\Delta} \mathbb{E}_{l\sim\mathbf{p}}\left[\phi_{t-1}^{\mathbf{b}}(\mathbf{s}+\mathbf{e}_l)\right] + \lambda\langle \mathbf{b}-\mathbf{p}, \mathbf{c}\rangle \text{ (Theorem 1)} \\
&= \min_{\mathbf{c} \in \mathcal{C}_0^{\text{eor}}} \max_{\mathbf{p}\in\Delta} \mathbb{E}_{l\sim\mathbf{p}}\left[\phi_{t-1}^{\mathbf{b}}(\mathbf{s}+\mathbf{e}_l)\right] + \langle \mathbf{b}-\mathbf{p}, \mathbf{c}\rangle \text{ (absorb } \lambda \text{ into } \mathbf{c}) \\
&= \max_{\mathbf{p}\in\Delta} \min_{\mathbf{c} \in \mathcal{C}_0^{\text{eor}}} \mathbb{E}_{l\sim\mathbf{p}}\left[\phi_{t-1}^{\mathbf{b}}(\mathbf{s}+\mathbf{e}_l)\right] + \langle \mathbf{b}-\mathbf{p}, \mathbf{c}\rangle \text{ (Theorem 1)}.
\end{aligned}$$

Unless $b(1) - p(1) \leq 0$ and $b(l) - p(l) \geq 0$ for each $l > 1$, the quantity $\langle \mathbf{b}-\mathbf{p}, \mathbf{c}\rangle$ can be made arbitrarily small for appropriate choices of $\mathbf{c} \in \mathcal{C}_0^{\text{eor}}$. The max-player is therefore forced to constrain its choices of $\mathbf{p}$, and the above expression becomes

$$\begin{array}{cc} \max\limits_{\mathbf{p}\in\Delta} & \mathbb{E}_{l\sim\mathbf{p}}\left[\phi_{t-1}^{\mathbf{b}}(\mathbf{s}+\mathbf{e}_l)\right] \\ \text{s.t.} & b(l) - q(l) \begin{cases} \geq 0 & \text{if } l = 1, \\ \leq 0 & \text{if } l > 1. \end{cases} \end{array}$$

Lemma 6 of Schapire (2001) states that if L is *proper* (as defined here), so is $\phi_t^{\mathbf{b}}$; the same result can be extended to our drifting games. This implies the optimal choice of $\mathbf{p}$ in the above expression is in fact the distribution that puts as small weight as possible in the first coordinate, namely $\mathbf{b}$. Therefore the optimum choice of $\mathbf{p}$ is $\mathbf{b}$, and the potential is the same as in (27).

We end the proof by showing that the choice of cost matrix in (28) is optimum. Theorem 9 states that a cost matrix $\mathbf{C}_t$ is the optimum choice if it satisfies (22), that is, if the expression

$$\max_{l=1}^{k}\left\{\phi_{T-t-1}^{\mathbf{B}(i)}(\mathbf{s}+\mathbf{e}_l) - \left(C_t(i,l) - \langle \mathbf{C}_t(i), \mathbf{B}(i)\rangle\right)\right\} \tag{29}$$

is equal to

$$\min_{\mathbf{c}\in\mathcal{C}_0} \max_{l=1}^{k}\left\{\phi_{T-t-1}^{\mathbf{B}(i)}(\mathbf{s}+\mathbf{e}_l) - \left(c(l) - \langle \mathbf{c}, \mathbf{B}(i)\rangle\right)\right\} = \phi_{T-t}^{\mathbf{B}(i)}(\mathbf{s}), \tag{30}$$

where the equality in (30) follows from (21). If $\mathbf{C}_t(i)$ is chosen as in (28), then, for any label l, the expression within max in (29) evaluates to

$$\begin{aligned}\phi^{\mathbf{B}(i)}_{T-t-1}(\mathbf{s}+\mathbf{e}_l) &- \left(\phi^{\mathbf{B}(i)}_{T-t-1}(\mathbf{s}+\mathbf{e}_l) - \langle \mathbf{C}_t(i), \mathbf{B}(i)\rangle\right) \\ &= \langle \mathbf{B}(i), \mathbf{C}_t(i)\rangle \\ &= \mathbb{E}_{l\sim\mathbf{B}(i)}\left[C_t(i,l)\right] \\ &= \mathbb{E}_{l\sim\mathbf{B}(i)}\left[\phi^{\mathbf{B}(i)}_{T-t-1}(\mathbf{s}+\mathbf{e}_l)\right] \\ &= \phi^{\mathbf{B}(i)}_{T-t}(\mathbf{s}),\end{aligned}$$

where the last equality follows from (27). Therefore the max expression in (29) is also equal to $\phi^{\mathbf{B}(i)}_{T-t}(\mathbf{s})$, which is what we needed to show. ■

Equation (28) in Lemma 10 implies the cost matrix chosen by the OS strategy can be expressed in terms of the potentials, which is the only thing left to calculate. Fortunately, the simplification (27) of the drifting games recurrence, allows the potentials to be solved completely in terms of a random-walk $\mathcal{R}^t_{\mathbf{b}}(\mathbf{x})$. This random variable denotes the position of a particle after t time steps, that starts at location $\mathbf{x} \in \mathbb{R}^k$, and in each step moves in direction $\mathbf{e}_l$ with probability $b(l)$.

Corollary 11 *The recurrence in* (27) *can be solved as follows:*

$$\phi^{\mathbf{b}}_t(\mathbf{s}) = \mathbb{E}\left[L\left(\mathcal{R}^t_{\mathbf{b}}(\mathbf{s})\right)\right]. \tag{31}$$

Proof Inductively assuming $\phi^{\mathbf{b}}_{t-1}(\mathbf{x}) = \mathbb{E}\left[L(\mathcal{R}^{t-1}_{\mathbf{b}}(\mathbf{x}))\right]$,

$$\phi_t(\mathbf{s}) = \mathbb{E}_{l\sim\mathbf{b}}\left[L(\mathcal{R}^{t-1}_{\mathbf{b}}(\mathbf{s})+\mathbf{e}_l)\right] = \mathbb{E}\left[L(\mathcal{R}^t_{\mathbf{b}}(\mathbf{s}))\right].$$

The last equality follows by observing that the random position $\mathcal{R}^{t-1}_{\mathbf{b}}(\mathbf{s})+\mathbf{e}_l$ is distributed as $\mathcal{R}^t_{\mathbf{b}}(\mathbf{s})$ when l is sampled from $\mathbf{b}$. ■

Lemma 10 and Corollary 11 together imply:

Theorem 12 *Assume L is proper and $\mathbf{b} \in \Delta^k_\gamma$ is an edge-over-random distribution. Then the potential $\phi^{\mathbf{b}}_t$, defined by the recurrence in* (17)*, has the solution given in* (31) *in terms of random walks.*

Before we can compute (31), we need to choose a loss function L. We next consider two options for the loss—the non-convex 0-1 error, and exponential loss.

6.1 Exponential Loss

The exponential loss serves as a smooth convex proxy for discontinuous non-convex 0-1 error (16) that we would ultimately like to bound, and is given by

$$L^{\exp}_\eta(\mathbf{s}) = \sum_{l=2}^{k} e^{\eta(s_l - s_1)}. \tag{32}$$

The parameter η can be thought of as the weight in each round, that is, $\alpha_t = \eta$ in each round. Then notice that the weighted state $\mathbf{f}_t$ of the examples, defined in (19), is related to the unweighted states

$\mathbf{s}_t$ as $f_t(l) = \eta s_t(l)$. Therefore the exponential loss function in (32) directly measures the loss of the weighted state as

$$L^{\exp}(\mathbf{f}_t) = \sum_{l=2}^{k} e^{f_t(l) - f_t(1)}. \tag{33}$$

Because of this correspondence, the optimal strategy with the loss function $L^{\exp}$ and $\alpha_t = \eta$ is the same as that using loss $L^{\exp}_{\eta}$ and $\alpha_t = 1$. We study the latter setting so that we may use the results derived earlier. With the choice of the exponential loss $L^{\exp}_{\eta}$, the potentials are easily computed, and in fact have a closed form solution.

Theorem 13 *If $L^{\exp}_{\eta}$ is as in* (32), *where η is non-negative, then the solution in Theorem 12 evaluates to $\phi_t^{\mathbf{b}}(\mathbf{s}) = \sum_{l=2}^{k} (a_l)^t e^{\eta_l (s_l - s_1)}$, where $a_l = 1 - (b_1 + b_l) + e^{\eta} b_l + e^{-\eta} b_1$.*

The proof by induction is straightforward. By tuning the weight η, each a_l can be always made less than 1. This ensures the exponential loss decays exponentially with rounds. In particular, when $\mathbf{B} = \mathbf{U}_{\gamma}$ (so that the condition is $(\mathcal{C}^{\mathrm{eor}}, \mathbf{U}_{\gamma})$), the relevant potential $\phi_t(\mathbf{s})$ or $\phi_t(\mathbf{f})$ is given by

$$\phi_t(\mathbf{s}) = \phi_t(f) = \kappa(\gamma, \eta)^t \sum_{l=2}^{k} e^{\eta(s_l - s_1)} = \kappa(\gamma, \eta)^t \sum_{l=2}^{k} e^{f_l - f_1} \tag{34}$$

where

$$\kappa(\gamma, \eta) = 1 + \frac{(1-\gamma)}{k} \left(e^{\eta} + e^{-\eta} - 2\right) - \left(1 - e^{-\eta}\right)\gamma. \tag{35}$$

The cost-matrix output by the OS algorithm can be simplified by rescaling, or adding the same number to each coordinate of a cost vector, without affecting the constraints it imposes on a weak classifier, to the following form

$$C(i,l) = \begin{cases} \left(e^{\eta} - 1\right) e^{\eta(s_l - s_1)} & \text{if } l > 1, \\ \left(e^{-\eta} - 1\right) \sum_{l=2}^{k} e^{\eta(s_l - s_1)} & \text{if } l = 1. \end{cases}$$

Using the correspondence between unweighted and weighted states, the above may also be rewritten as:

$$C(i,l) = \begin{cases} \left(e^{\eta} - 1\right) e^{f_l - f_1} & \text{if } l > 1, \\ \left(e^{-\eta} - 1\right) \sum_{l=2}^{k} e^{f_l - f_1} & \text{if } l = 1. \end{cases} \tag{36}$$

With such a choice, Theorem 9 and the form of the potential guarantee that the average loss

$$\frac{1}{m} \sum_{i=1}^{m} L^{\exp}_{\eta}(\mathbf{s}_t(i)) = \frac{1}{m} \sum_{i=1}^{m} L^{\exp}(\mathbf{f}_t(i)) \tag{37}$$

of the states changes by a factor of at most $\kappa(\gamma, \eta)$ every round. Therefore the final loss, which upper bounds the error, that is, the fraction of misclassified training examples, is at most $(k-1)\kappa(\gamma, \eta)^T$. Since this upper bound holds for any value of η, we may tune it to optimize the bound. Setting $\eta = \ln(1+\gamma)$, the error can be upper bounded by $(k-1)e^{-T\gamma^2/2}$.

6.2 Zero-one Loss

There is no simple closed form solution for the potential when using the zero-one loss L^{err} (16). However, we may compute the potentials efficiently as follows. To compute $\phi_t^{\mathbf{b}}(\mathbf{s})$, we need to find the probability that a random walk (making steps according to $\mathbf{b}$) of length t in $\mathbb{Z}^k$, starting at $\mathbf{s}$ will end up in a region where the loss function is 1. Any such random walk will consist of x_l steps in direction $\mathbf{e}_l$ where the non-negative $\sum_l x_l = t$. The probability of each such path is $\prod_l b_l^{x_l}$. Further, there are exactly $\binom{t}{x_1,\ldots,x_k}$ such paths. Starting at state $\mathbf{s}$, such a path will lead to a correct answer only if $s_1 + x_1 > s_l + x_l$ for each $l > 1$. Hence we may write the potential $\phi_t^{\mathbf{b}}(\mathbf{s})$ as

$$\begin{array}{rlcl} \phi_t^{\mathbf{b}}(\mathbf{s}) = 1 - \displaystyle\sum_{x_1,\ldots,x_k}^{t} & \binom{t}{x_1,\ldots,x_k} \prod_{l=1}^k b_l^{x_l} & & \\ \text{s.t.} & x_1 + \ldots + x_k & = t \\ \forall l : & x_l & \geq 0 \\ \forall l : & x_l + s_l & \leq x_1 + s_1 . \end{array}$$

Since the x_l's are restricted to be integers, this problem is presumably hard. In particular, the only algorithms known to the authors that take time logarithmic in t is also exponential in k. However, by using dynamic programming, we can compute the summation in time polynomial in $|s_l|$, t and k. In fact, the run time is always $O(t^3 k)$, and at least $\Omega(tk)$.

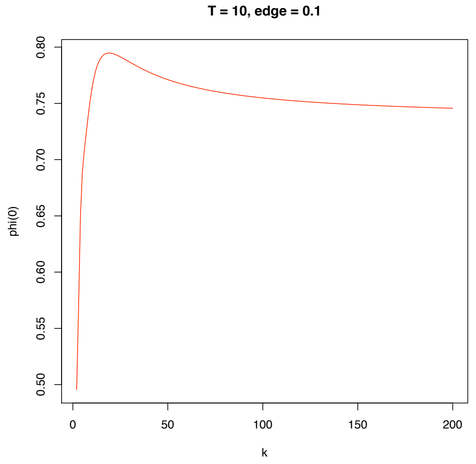
The bounds on error we achieve, although not in closed form, are much tighter than those obtainable using exponential loss. The exponential loss analysis yields an error upper bound of $(k-1)e^{-T\gamma^2/2}$. Using a different initial distribution, Schapire and Singer (1999) achieve the slightly better bound $\sqrt{(k-1)}e^{-T\gamma^2/2}$. However, when the edge γ is small and the number of rounds are few, each bound is greater than 1 and hence trivial. On the other hand, the bounds computed by the above dynamic program are sensible for all values of k, γ and T. When $\mathbf{b}$ is the γ-biased uniform distribution $\mathbf{b} = (\frac{1-\gamma}{k} + \gamma, \frac{1-\gamma}{k}, \frac{1-\gamma}{k}, \ldots, \frac{1-\gamma}{k})$ a table containing the error upper bound $\phi_T^{\mathbf{b}}(0)$ for $k = 6$, $\gamma = 0$ and small values for the number of rounds T is shown in Figure 2(a); note that with the exponential loss, the bound is always 1 if the edge γ is 0. Further, the bounds due to the exponential loss analyses seem to imply that the dependence of the error on the number of labels is monotonic. However, a plot of the tighter bounds with edge $\gamma = 0.1$, number of rounds $T = 10$ against various values of k, shown in Figure 2(b), indicates that the true dependence is more complicated. Therefore the tighter analysis also provides qualitative insights not obtainable via the exponential loss bound.

7. Solving for the Minimal Weak Learning Condition

In the previous section we saw how to find the optimal boosting strategy when using any fixed edge-over-random condition. However as we have seen before, these conditions can be stronger than necessary, and therefore lead to boosting algorithms that require additional assumptions. Here we show how to compute the optimal algorithm while using the weakest weak learning condition, provided by (13), or equivalently the condition used by AdaBoost.MR, $(\mathcal{C}^{\text{MR}}, \mathbf{B}_\gamma^{\text{MR}})$. Since there are two possible formulations for the minimal condition, it is not immediately clear which to use to compute the optimal boosting strategy. To resolve this, we first show that the optimal boosting strategy based on any formulation of a necessary and sufficient weak learning condition is the same. Having resolved this ambiguity, we show how to compute this strategy for the exponential loss and 0-1 error using the first formulation.

T	$\phi_T^{\mathbf{b}}(\mathbf{0})$	T	$\phi_T^{\mathbf{b}}(\mathbf{0})$
0	1.00	6	0.90
1	0.83	7	0.91
2	0.97	8	0.90
3	0.93	9	0.89
4	0.89	10	0.89
5	0.89		

(a)



(b)

Figure 2: Plot of potential value $\phi_T^{\mathbf{b}}(\mathbf{0})$ where $\mathbf{b}$ is the γ-biased uniform distribution: $\mathbf{b} = (\frac{1-\gamma}{k} + \gamma, \frac{1-\gamma}{k}, \frac{1-\gamma}{k}, \ldots, \frac{1-\gamma}{k})$. **(a):** Potential values (rounded to two decimal places) for different number of rounds T using $\gamma = 0$ and $k = 6$. These are bounds on the error, and less than 1 even when the edge and number of rounds are small. **(b):** Potential values for different number of classes k, with $\gamma = 0.1$, and $T = 10$. These are tight estimates for the optimal error, and yet not monotonic in the number of classes.

7.1 Game-theoretic Equivalence of Necessary and Sufficient Weak-learning Conditions

In this section we study the effect of the weak learning condition on the game-theoretically optimal boosting strategy. We introduce the notion of *game-theoretic equivalence* between two weak learning conditions, that determines if the payoffs (15) of the optimal boosting strategies based on the two conditions are identical. This should hold whenever both games last for the same number of iterations T, for any value of T. This is different from the usual notion of equivalence between two conditions, which holds if any weak classifier space satisfies both conditions or neither condition. In fact we prove that game-theoretic equivalence is a broader notion; in other words, equivalence implies game-theoretic equivalence. A special case of this general result is that any two weak learning conditions that are necessary and sufficient, and hence equivalent to boostability, are also game-theoretically equivalent. In particular, so are the conditions of AdaBoost.MR and (13), and the resulting optimal Booster strategies enjoy equally good payoffs. We conclude that in order to derive the optimal boosting strategy that uses the minimal weak-learning condition, it is sound to use either of these two formulations.

The purpose of a weak learning condition $(\mathcal{C}, \mathbf{B})$ is to impose restrictions on the Weak-Learner's responses in each round. These restrictions are captured by subsets of the weak classifier space as follows. If Booster chooses cost-matrix $\mathbf{C} \in \mathcal{C}$ in a round, the Weak-Learner's response h is restricted to the subset $S_{\mathbf{C}} \subseteq \mathcal{H}^{\text{all}}$ defined as

$$S_{\mathbf{C}} = \left\{ h \in \mathcal{H}^{\text{all}} : \mathbf{C} \bullet \mathbf{1}_h \leq \mathbf{C} \bullet \mathbf{B} \right\}.$$

Thus, a weak learning condition is essentially a family of subsets of the weak classifier space. Further, smaller subsets mean fewer options for Weak-Learner, and hence better payoffs for the optimal boosting strategy. Based on this idea, we may define when a weak learning condition $(\mathcal{C}_1, \mathbf{B}_1)$ is *game-theoretically stronger* than another condition $(\mathcal{C}_2, \mathbf{B}_2)$ if the following holds: For every subset $S_{\mathbf{C}_2}$ in the second condition (that is $\mathbf{C}_2 \in \mathcal{C}_2$), there is a subset $S_{\mathbf{C}_1}$ in the first condition (that is $\mathbf{C}_1 \in \mathcal{C}_1$), such that $S_{\mathbf{C}_1} \subseteq S_{\mathbf{C}_2}$. Mathematically, this may be written as follows:

$$\forall \mathbf{C}_1 \in \mathcal{C}_1, \exists \mathbf{C}_2 \in \mathcal{C}_2 : S_{\mathbf{C}_1} \subseteq S_{\mathbf{C}_2}.$$

Intuitively, a game theoretically stronger condition will allow Booster to place similar or stricter restrictions on Weak-Learner in each round. Therefore, the optimal Booster payoff using a game-theoretically stronger condition is at least equally good, if not better. It therefore follows that if two conditions are both game-theoretically stronger than each other, the corresponding Booster payoffs must be equal, that is they must be *game-theoretically equivalent*.

Note that game-theoretic equivalence of two conditions does not mean that they are identical as families of subsets, for we may arbitrarily add large and "useless" subsets to the two conditions without affecting the Booster payoffs, since these subsets will never be used by an optimal Booster strategy. In fact we next show that game-theoretic equivalence is a broader notion than just equivalence.

Theorem 14 *Suppose $(\mathcal{C}_1, \mathbf{B}_1)$ and $(\mathcal{C}_2, \mathbf{B}_2)$ are two equivalent weak learning conditions, that is, every space $\mathcal{H}$ satisfies both or neither condition. Then each condition is game-theoretically stronger than the other, and hence game-theoretically equivalent.*

Proof We argue by contradiction. Assume that despite equivalence, the first condition (without loss of generality) includes a particularly hard subset $S_{\mathbf{C}_1} \subseteq \mathcal{H}^{\text{all}}, \mathbf{C}_1 \in \mathcal{C}_1$ which is not smaller than any subset in the second condition. In particular, for every subset $S_{\mathbf{C}_2}, \mathbf{C}_2 \in \mathcal{C}_2$ in the second condition is satisfied by some weak classifier $h_{\mathbf{C}_2}$ not satisfying the hard subset in the first condition: $h_{\mathbf{C}_2} \in S_{\mathbf{C}_2} \setminus S_{\mathbf{C}_1}$. Therefore, the space

$$\mathcal{H} = \{h_{\mathbf{C}_2} : \mathbf{C}_2 \in \mathcal{C}_2\},$$

formed by just these classifiers satisfies the second condition, but has an empty intersection with $S_{\mathbf{C}_1}$. In other words, $\mathcal{H}$ satisfies the second but not the first condition, a contradiction to their equivalence. ■

An immediate corollary is the game theoretic equivalence of necessary and equivalent conditions.

Corollary 15 *Any two necessary and sufficient weak learning conditions are game-theoretically equivalent. In particular the optimum Booster strategies based on AdaBoost.MR's condition $(\mathcal{C}^{MR}, \mathbf{B}_\gamma^{MR})$ and* (13) *have equal payoffs.*

Therefore, in deriving the optimal Booster strategy, it is sound to work with either AdaBoost.MR's condition or (13). In the next section, we actually compute the optimal strategy using the latter formulation.

7.2 Optimal Strategy with the Minimal Conditions

In this section we compute the optimal Booster strategy that uses the minimum weak learning condition provided in (13). We choose this instead of AdaBoost.MR's condition because this description is more closely related to the edge-over-random conditions, and the resulting algorithm has a close relationship to the ones derived for fixed edge-over-random conditions, and therefore more insightful. However, this formulation does not state the condition as a single pair $(\mathbf{C}, \mathbf{B})$, and therefore we cannot directly use the result of Theorem 9. Instead, we define new potentials and a modified OS strategy that is still nearly optimal, and this constitutes the first part of this section. In the second part, we show how to compute these new potentials and the resulting OS strategy.

7.2.1 MODIFIED POTENTIALS AND OS STRATEGY

The condition in (13) is not stated as a single pair $(\mathcal{C}^{\text{eor}}, \mathbf{B})$, but uses all possible edge-over-random baselines $\mathbf{B} \in \mathcal{B}_\gamma^{\text{eor}}$. Therefore, we modify the definitions (17) of the potentials accordingly to extract an optimal Booster strategy. Recall that Δ_γ^k is defined in (26) as the set of all edge-over-random distributions which constitute the rows of edge-over-random baselines $\mathbf{B} \in \mathcal{B}_\gamma^{\text{eor}}$. Using these, define new potentials $\phi_t(\mathbf{s})$ as follows:

$$\phi_t(\mathbf{s}) = \begin{array}{ccc} \min\limits_{\mathbf{c} \in \mathcal{C}_0^{\text{eor}}} & \max\limits_{\mathbf{b} \in \Delta_\gamma^k} \max\limits_{\mathbf{p} \in \Delta\{1,\ldots,k\}} & \mathbb{E}_{l \sim \mathbf{p}} \left[\phi_{t-1}\left(\mathbf{s} + \mathbf{e}_l\right)\right] \\ & \text{s.t.} & \mathbb{E}_{l \sim \mathbf{p}}[c(l)] \leq \langle \mathbf{b}, \mathbf{c} \rangle . \end{array} \tag{38}$$

The main difference between (38) and (17) is that while the older potentials were defined using a fixed vector $\mathbf{b}$ corresponding to some row in the fixed baseline $\mathbf{B}$, the new definition takes the maximum over all possible rows $\mathbf{b} \in \Delta_\gamma^k$ that an edge-over-random baseline $\mathbf{B} \in \mathcal{B}_\gamma^{\text{eor}}$ may have. As before, we may write the recurrence in (38) in its dual form

$$\phi_t(\mathbf{s}) = \min_{\mathbf{c} \in \mathcal{C}_0^{\text{eor}}} \max_{\mathbf{b} \in \Delta_\gamma^k} \max_{l=1}^{k} \left\{\phi_{t-1}\left(\mathbf{s} + \mathbf{e}_l\right) - \left(c(l) - \langle \mathbf{c}, \mathbf{b} \rangle\right)\right\}.$$

The proof is very similar to that of Lemma 8 and is omitted. We may now define a new OS strategy that chooses a cost-matrix in round t analogously:

$$\mathbf{C}_t(i) \in \operatorname*{argmin}_{\mathbf{c} \in \mathcal{C}_0^{\text{eor}}} \max_{\mathbf{b} \in \Delta_\gamma^k} \max_{l=1}^{k} \left\{\phi_{t-1}\left(\mathbf{s} + \mathbf{e}_l\right) - \left(c(l) - \langle \mathbf{c}, \mathbf{b} \rangle\right)\right\}. \tag{39}$$

where recall that $\mathbf{s}_t(i)$ denotes the state vector (defined in (18)) of example i. With this strategy, we can show an optimality result very similar to Theorem 9.

Theorem 16 *Suppose the weak-learning condition is given by* (13)*. Let the potential functions $\phi_t^{\mathbf{b}}$ be defined as in* (38)*, and assume the Booster employs the modified OS strategy, choosing $\alpha_t = 1$ and $\mathbf{C}_t$ as in* (39) *in each round t. Then the average potential of the states,*

$$\frac{1}{m} \sum_{i=1}^{m} \phi_{T-t}\left(\mathbf{s}_t(i)\right),$$

never increases in any round. In particular, the loss suffered after T rounds of play is at most $\phi_T(\mathbf{0})$.

Further, for any $\varepsilon > 0$, when the loss function satisfies (24) *and the number of examples m is as large as in* (25)*, no Booster strategy can guarantee to achieve less than $\phi_T(\mathbf{0}) - \varepsilon$ loss in T rounds.*

The proof is very similar to that of Theorem 9 and is omitted.

7.2.2 COMPUTING THE NEW POTENTIALS

Here we show how to compute the new potentials. The resulting algorithms will require exponential time, and we provide some empirical evidence showing that this might be necessary. Finally, we show how to carry out the computations efficiently in certain special situations.
An Exponential Time Algorithm. Here we show how the potentials may be computed as the expected loss of some random walk, just as we did for the potentials arising with fixed edge-over-random conditions. The main difference is there will be several random walks to choose from.

We first begin by simplifying the recurrence (38), and expressing the optimal cost matrix in (39) in terms of the potentials, just as we did in Lemma 10 for the case of fixed edge-over-random conditions.

Lemma 17 *Assume L is proper. Then the recurrence* (38) *may be simplified as*

$$\phi_t(\mathbf{s}) \quad = \quad \max_{\mathbf{b}\in\Delta_\gamma^k} \mathbb{E}_{l\sim\mathbf{b}}\left[\phi_{t-1}\left(\mathbf{s}+\mathbf{e}_l\right)\right]. \tag{40}$$

Further, if the cost matrix $\mathbf{C}_t$ *is chosen as follows:*

$$C_t(i,l) = \phi_{T-t-1}(\mathbf{s}_t(i)+\mathbf{e}_l), \tag{41}$$

then $\mathbf{C}_t$ *satisfies the condition in* (39).

The proof is very similar to that of Lemma 10 and is omitted. Equation (41) implies that, as before, computing the optimal Booster strategy reduces to computing the new potentials. One computational difficulty created by the new definitions (38) or (40) is that they require infinitely many possible distributions $\mathbf{b} \in \Delta_\gamma^k$ to be considered. We show that we may in fact restrict our attention to only finitely many of such distributions described next.

At any state $\mathbf{s}$ and number of remaining iterations t, let π be a permutation of the coordinates $\{2,\ldots,k\}$ that sorts the potential values:

$$\phi_{t-1}\left(\mathbf{s}+\mathbf{e}_{\pi(k)}\right) \geq \phi_{t-1}\left(\mathbf{s}+\mathbf{e}_{\pi(k-1)}\right) \geq \ldots \geq \phi_{t-1}\left(\mathbf{s}+\mathbf{e}_{\pi(2)}\right). \tag{42}$$

For any permutation π of the coordinates $\{2,\ldots,k\}$, let $\mathbf{b}_a^\pi$ denote the γ-biased uniform distribution on the a coordinates $\{1,\pi_k,\pi_{k-1},\ldots,\pi_{k-a+2}\}$:

$$b_a^\pi(l) = \begin{cases} \frac{1-\gamma}{a}+\gamma & \text{if } l=1 \\ \frac{1-\gamma}{a} & \text{if } l\in\{\pi_k,\ldots,\pi_{k-a+2}\} \\ 0 & \text{otherwise.} \end{cases} \tag{43}$$

Then, the next lemma shows that we may restrict our attention to only the distributions $\left\{\mathbf{b}_2^\pi,\ldots,\mathbf{b}_k^\pi\right\}$ when evaluating the recurrence in (40).

Lemma 18 *Fix a state* $\mathbf{s}$ *and remaining rounds of boosting t. Let* π *be a permutation of the coordinates* $\{2,\ldots,k\}$ *satisfying* (42)*, and define* $\mathbf{b}_a^\pi$ *as in* (43). *Then the recurrence* (40) *may be simplified as follows:*

$$\phi_t(\mathbf{s}) = \max_{\mathbf{b}\in\Delta_\gamma^k} \mathbb{E}_{l\sim\mathbf{b}}\left[\phi_{t-1}\left(\mathbf{s}+\mathbf{e}_l\right)\right] = \max_{2\leq a\leq k} \mathbb{E}_{l\sim\mathbf{b}_a^\pi}\left[\phi_{t-1}\left(\mathbf{s}+\mathbf{e}_l\right)\right]. \tag{44}$$

Proof Assume (by relabeling the coordinates if necessary) that π is the identity permutation, that is, $\pi(2) = 2, \ldots, \pi(k) = k$. Observe that the right hand side of (40) is at least as much the right hand side of (44) since the former considers more distributions. We complete the proof by showing that the former is also at most the latter.

By (40), we may assume that some optimal $\mathbf{b}$ satisfies

$$\begin{aligned} b(k) = \cdots = b(k-a+2) &= b(1) - \gamma, \\ b(k-a+1) &\le b(1) - \gamma, \\ b(k-a) = \cdots = b(2) &= 0. \end{aligned}$$

Therefore, $\mathbf{b}$ is a distribution supported on $a+1$ elements, with the minimum weight placed on element $k-a+1$. This implies $b(k-a+1) \in [0, 1/(a+1)]$.

Now, $\mathbb{E}_{l\sim\mathbf{b}}[\phi_{t-1}(\mathbf{s}+\mathbf{e}_l)]$ may be written as

$$\begin{aligned} & \gamma \cdot \phi_{t-1}(\mathbf{s}+\mathbf{e}_1) + b(k-a+1)\phi_{t-1}(\mathbf{s}+\mathbf{e}_{k-a+1}) \\ + \quad & (1-\gamma-b(k-a+1))\frac{\phi_{t-1}(\mathbf{s}+\mathbf{e}_1) + \phi_{t-1}(\mathbf{s}+\mathbf{e}_{k-a+2}) + \ldots \phi_{t-1}(\mathbf{s}+\mathbf{e}_k)}{a} \\ = \quad & \gamma \cdot \phi_{t-1}(\mathbf{s}+\mathbf{e}_1) + \frac{b(k-a+1)}{1-\gamma}\phi_{t-1}(\mathbf{s}+\mathbf{e}_{k-a+1}) \\ + \quad & (1-\gamma)\left\{ \left(1 - \frac{b(k-a+1)}{1-\gamma}\right) \frac{\phi_{t-1}(\mathbf{s}+\mathbf{e}_1) + \phi_{t-1}(\mathbf{s}+\mathbf{e}_{k-a+2}) + \ldots \phi_{t-1}(\mathbf{s}+\mathbf{e}_k)}{a} \right\} \end{aligned}$$

Replacing $b(k-a+1)$ by x in the above expression, we get a linear function of x. When restricted to $[0, 1/(a+1)]$ the maximum value is attained at a boundary point. For $x = 0$, the expression becomes

$$\gamma \cdot \phi_{t-1}(\mathbf{s}+\mathbf{e}_1) + (1-\gamma)\frac{\phi_{t-1}(\mathbf{s}+\mathbf{e}_1) + \phi_{t-1}(\mathbf{s}+\mathbf{e}_{k-a+2}) + \ldots \phi_{t-1}(\mathbf{s}+\mathbf{e}_k)}{a}.$$

For $x = 1/(a+1)$, the expression becomes

$$\gamma \cdot \phi_{t-1}(\mathbf{s}+\mathbf{e}_1) + (1-\gamma)\frac{\phi_{t-1}(\mathbf{s}+\mathbf{e}_1) + \phi_{t-1}(\mathbf{s}+\mathbf{e}_{k-a+1}) + \ldots \phi_{t-1}(\mathbf{s}+\mathbf{e}_k)}{a+1}.$$

Since $b(k-a+1)$ lies in $[0, 1/(a+1)]$, the optimal value is at most the maximum of the two. However each of these last two expressions is at most the right hand side of (44), completing the proof. ■

Unraveling (44), we find that $\phi_t(\mathbf{s})$ is the expected loss of the final state reached by some random walk of t steps starting at state $\mathbf{s}$. However, the number of possibilities for the random-walk is huge; indeed, the distribution at each step can be any of the $k-1$ possibilities $\mathbf{b}_a^\pi$ for $a \in \{2, \ldots, k\}$, where the parameter a denotes the size of the support of the γ-biased uniform distribution chosen at each step. In other words, at a given state $\mathbf{s}$ with t rounds of boosting remaining, the parameter a determines the number of directions the optimal random walk will consider taking; we will therefore refer to a as the *degree* of the random walk given $(\mathbf{s}, t)$. Now, the total number of states reachable in T steps is $O\left(T^{k-1}\right)$. If the degree assignment every such state, for every value of $t \le T$ is fixed in advance, $\mathbf{a} = \{a(\mathbf{s}, t) : t \le T, \mathbf{s} \text{ reachable}\}$, we may identify a unique random walk $\mathcal{R}^{\mathbf{a},t}(\mathbf{s})$ of length t starting at step $\mathbf{s}$. Therefore the potential may be computed as

$$\phi_t(\mathbf{s}) = \max_{\mathbf{a}} \mathbb{E}\left[\mathcal{R}^{\mathbf{a},t}(\mathbf{s})\right]. \tag{45}$$

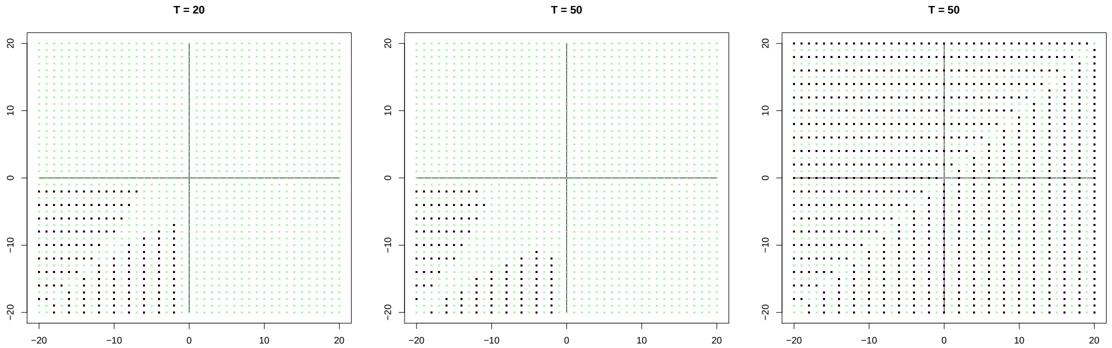


Figure 3: Green pixels (crosses) have degree 3, black pixels (solid squares) have degree 2. Each step is diagonally down (left), and up (if $x < y$) and right (if $x > y$) and both when degree is 3. The rightmost figure uses $\gamma = 0.4$, and the other two $\gamma = 0$. The loss function is 0-1.

A dynamic programming approach for computing (45) requires time and memory linear in the number of different states reachable by a random walk that takes T coordinate steps: $O(T^{k-1})$. This is exponential in the data set size, and hence impractical. In the next two sections we show that perhaps there may not be any way of computing these efficiently in general, but provide efficient algorithms in certain special cases.

Hardness of Evaluating the Potentials. Here we provide empirical evidence for the hardness of computing the new potentials. We first identify a computationally easier problem, and show that even that is probably hard to compute. Equation (44) implies that if the potentials were efficiently computable, the correct value of the degree a could be determined efficiently. The problem of determining the degree a given the state $\mathbf{s}$ and remaining rounds t is therefore easier than evaluating the potentials. However, a plot of the degrees against states and remaining rounds, henceforth called a *degree map*, reveals very little structure that might be captured by a computationally efficient function.

We include three such degree maps in Figure 3. Only three classes $k = 3$ are used, and the loss function is 0-1 error. We also fix the number T of remaining rounds of boosting and the value of the edge γ for each plot. For ease of presentation, the 3-dimensional states $\mathbf{s} = (s_1, s_2, s_3)$ are compressed into 2-dimensional pixel coordinates $(u = s_2 - s_1, v = s_3 - s_2)$. It can be shown that this does not take away information required to evaluate the potentials or the degree at any pixel (u, v). Further, only those states are considered whose compressed coordinates u, v lie in the range $[-T, T]$; in T rounds, these account for all the reachable states. The degrees are indicated on the plot by colors. Our discussion in the previous sections implies that the possible values of the degree is 2 or 3. When the degree at a pixel (u, v) is 3, the pixel is colored green, and when the degree is 2, it is colored black.

Note that a random walk over the space $\mathbf{s} \in \mathbb{R}^3$ consisting of distributions over coordinate steps $\{(1,0,0), (0,1,0), (0,0,1)\}$ translates to a random walk over $(u, v) \in \mathbb{R}^2$ where each step lies in the set $\{(-1,-1), (1,0), (0,1)\}$. In Figure 3, these correspond to the directions diagonally down, up or right. Therefore at a black pixel, the random walk either chooses between diagonally down and up, or between diagonally down and right, with probabilities $\{1/2 + \gamma/2, 1/2 - \gamma/2\}$. On the

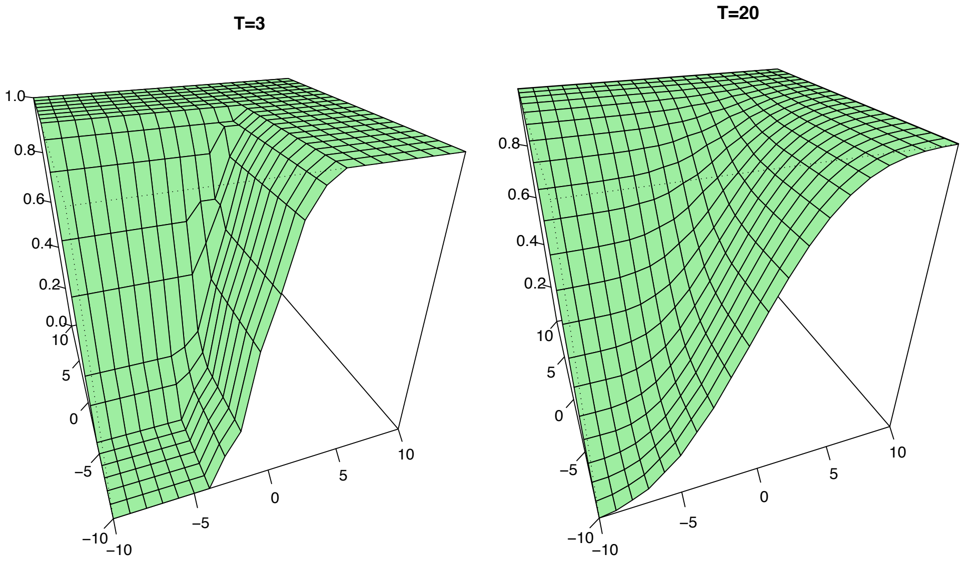


Figure 4: Optimum recurrence value. We set $\gamma = 0$. Surface is irregular for smaller values of T, but smoother for larger values, admitting hope for approximation.

other hand, at a green pixel, the random walk chooses among diagonally down, up and right with probabilities $(\gamma + (1-\gamma)/3, (1-\gamma)/3, (1-\gamma)/3)$. The degree maps are shown for varying values of T and the edge γ. While some patterns emerge for the degrees, such as black or green depending on the parity of u or v, the authors found the region near the line $u = v$ still too complex to admit any solution apart from a brute-force computation.

We also plot the potential values themselves in Figure 4 against different states. In each plot, the number of iterations remaining, T, is held constant, the number of classes is chosen to be 3, and the edge $\gamma = 0$. The states are compressed into pixels as before, and the potential is plotted against each pixel, resulting in a 3-dimensional surface. We include two plots, with different values for T. The surface is irregular for $T = 3$ rounds, but smoother for 20 rounds, admitting some hope for approximation.

An alternative approach would be to approximate the potential ϕ_t by the potential $\phi_t^{\mathbf{b}}$ for some fixed $\mathbf{b} \in \Delta_\gamma^k$ corresponding to some particular edge-over-random condition. Since $\phi_t \geq \phi_t^{\mathbf{b}}$ for all edge-over-random distributions $\mathbf{b}$, it is natural to approximate by choosing $\mathbf{b}$ that maximizes the fixed edge-over-random potential. (It can be shown that this $\mathbf{b}$ corresponds to the γ-biased uniform distribution.) Two plots of comparing the potential values at $\mathbf{0}$, $\phi_T(\mathbf{0})$ and $\max_{\mathbf{b}} \phi_T^{\mathbf{b}}(\mathbf{0})$, which correspond to the respective error upper bounds, is shown in Figure 5. In the first plot, the number of classes k is held fixed at 6, and the values are plotted for different values of iterations T. In the second plot, the number of classes vary, and the number of iterations is held at 10. Both plots show that the difference in the values is significant, and hence $\max_{\mathbf{b}} \phi_T^{\mathbf{b}}(\mathbf{0})$ would be a rather optimistic upper bound on the error when using the minimal weak learning condition.

If we use exponential loss (32), the situation is not much better. The degree maps for varying values of the weight parameter η against fixed values of edge $\gamma = 0.1$, rounds remaining $T = 20$ and

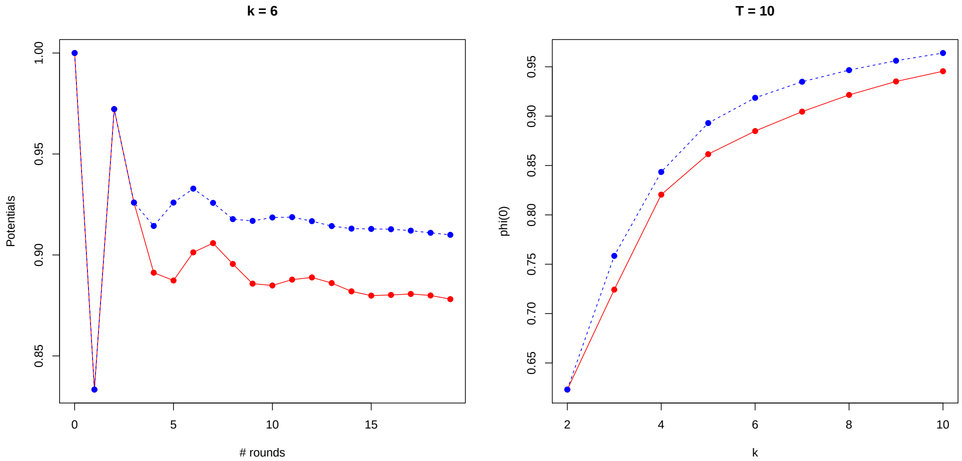


Figure 5: Comparison of $\phi_t(\mathbf{0})$ (blue, dashed) with $\max_{\mathbf{q}} \phi_t^{\mathbf{q}}(\mathbf{0})$ (red, solid) over different rounds t and different number of classes k. We set $\gamma = 0$ in both.

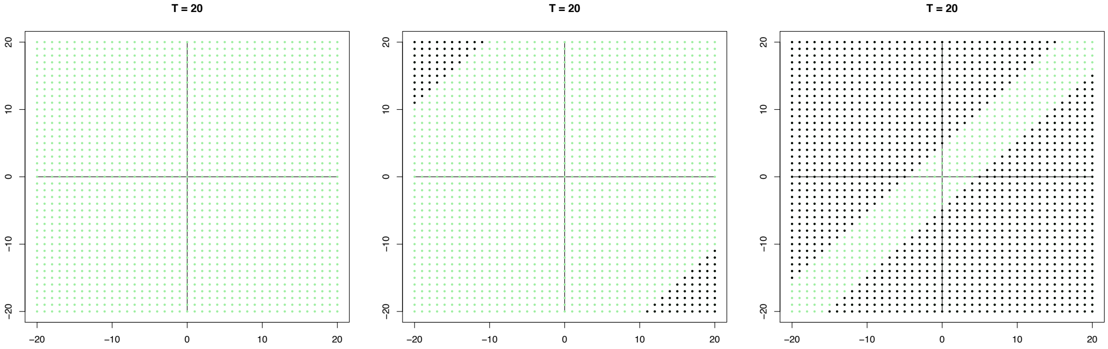


Figure 6: Green pixels (crosses) have degree 3, black pixels (squares) have degree 2. Each step is diagonally down (left), and up (if $x < y$) and right (if $x > y$) and both when degree is 3. Each plot uses $T = 20, \gamma = 0.1$. The values of η are 0.08, 0.1 and 0.3, respectively. With smaller values of η, more pixels have degree 3.

number of classes $k = 3$ are plotted in Figure 6. Although the patterns are simple, with the degree 3 pixels forming a diagonal band, we found it hard to prove this fact formally, or compute the exact boundary of the band. However the plots suggest that when η is small, all pixels have degree 3. We next find conditions under which this opportunity for tractable computation exists.

Efficient Computation in Special Cases. Here we show that when using the exponential loss, if the edge γ is very small, then the potentials can be computed efficiently. We first show an intermediate result. We already observed empirically that when the weight parameter η is small, the degrees all become equal to k. Indeed, we can prove this fact.

Lemma 19 *If the loss function being used is exponential loss* (32) *and the weight parameter* η *is small compared to the number of rounds*

$$\eta \leq \frac{1}{4}\min\left\{\frac{1}{k-1}, \frac{1}{T}\right\}, \tag{46}$$

then the optimal value of the degree a in (44) *is always k. Therefore, in this situation, the potential* ϕ_t *using the minimal weak learning condition is the same as the potential* $\phi_t^{\mathbf{u}}$ *using the* γ*-biased uniform distribution* $\mathbf{u}$*,*

$$\mathbf{u} = \left(\frac{1-\gamma}{k}+\gamma, \frac{1-\gamma}{k}, \ldots, \frac{1-\gamma}{k}\right), \tag{47}$$

and hence can be efficiently computed.

Proof We show $\phi_t = \phi_t^{\mathbf{u}}$ by induction on the remaining number t of boosting iterations. The base case holds since, by definition, $\phi_0 = \phi_0^{\mathbf{u}} = L_\eta^{\exp}$. Assume, inductively that

$$\phi_{t-1}(\mathbf{s}) = \phi_{t-1}^{\mathbf{u}}(\mathbf{s}) = \kappa(\gamma,\eta)^{t-1}\sum_{l=2}^{k} e^{\eta(s_l - s_1)}, \tag{48}$$

where the second equality follows from (34). We show that

$$\phi_t(\mathbf{s}) = \mathbb{E}_{l\sim\mathbf{u}}\left[\phi_{t-1}(\mathbf{s}+\mathbf{e}_l)\right]. \tag{49}$$

By the inductive hypothesis and (27), the right hand side of (49) is in fact equal to $\phi_t^{\mathbf{u}}$, and we will have shown $\phi_t = \phi_t^{\mathbf{u}}$. The proof will then follow by induction.

In order to show (49), by Lemma 18, it suffices to show that the optimal degree a maximizing the right hand side of (44) is always k:

$$\mathbb{E}_{l\sim\mathbf{b}_a^\pi}\left[\phi_{t-1}\left(\mathbf{s}+\mathbf{e}_l\right)\right] \leq \mathbb{E}_{l\sim\mathbf{b}_k^\pi}\left[\phi_{t-1}\left(\mathbf{s}+\mathbf{e}_l\right)\right]. \tag{50}$$

By (48), $\phi_{t-1}\left(\mathbf{s}+\mathbf{e}_{l_0}\right)$ may be written as $\phi_{t-1}(\mathbf{s}) + \kappa(\gamma,\eta)^{t-1}\cdot\xi_{l_0}$, where the term ξ_{l_0} is:

$$\xi_{l_0} = \begin{cases} (e^\eta - 1)e^{\eta(s_{l_0}-s_1)} & \text{if } l_0 \neq 1, \\ (e^{-\eta}-1)\sum_{l=2}^k e^{\eta(s_l - s_1)} & \text{if } l_0 = 1. \end{cases}$$

Therefore (50) is the same as: $\mathbb{E}_{l\sim\mathbf{b}_a^\pi}[\xi_l] \leq \mathbb{E}_{l\sim\mathbf{b}_k^\pi}[\xi_l]$. Assume (by relabeling if necessary) that π is the identity permutation on coordinates $\{2,\ldots,k\}$. Then the expression $\mathbb{E}_{l\sim\mathbf{b}_a^\pi}[\xi_l]$ can be written as

$$\begin{aligned} \mathbb{E}_{l\sim\mathbf{b}_a^\pi}[\xi_l] &= \left(\frac{1-\gamma}{a}+\gamma\right)\xi_1 + \sum_{l=k-a+2}^{k}\left(\frac{1-\gamma}{a}\right)\xi_l \\ &= \gamma\xi_1 + (1-\gamma)\left\{\frac{\xi_1 + \sum_{l=k-a+2}^k \xi_l}{a}\right\}. \end{aligned}$$

It suffices to show that the term in curly brackets is maximized when $a = k$. We will in fact show that the numerator of the term is negative if $a < k$, and non-negative for $a = k$, which will complete our proof. Notice that the numerator can be written as

$$\begin{aligned}
& (e^{\eta}-1)\left\{\sum_{l=k-a+2}^{k} e^{\eta(s_l-s_1)}\right\}-(1-e^{-\eta})\sum_{l=2}^{k} e^{\eta(s_l-s_1)} \\
= \quad & (e^{\eta}-1)\left\{\sum_{l=k-a+2}^{k} e^{\eta(s_l-s_1)}-\sum_{l=2}^{k} e^{\eta(s_l-s_1)}\right\}+\left\{(e^{\eta}-1)-(1-e^{-\eta})\right\}\sum_{l=2}^{k} e^{\eta(s_l-s_1)} \\
= \quad & \left\{e^{\eta}+e^{-\eta}-2\right\}\sum_{l=2}^{k} e^{\eta(s_l-s_1)}-(e^{\eta}-1)\left\{\sum_{l=2}^{k-a+1} e^{\eta(s_l-s_1)}\right\}.
\end{aligned}$$

When $a=k$, the second summation disappears, and we are left with a non-negative expression. However when $a<k$, the second summation is at least $e^{\eta(s_2-s_1)}$. Since $t \leq T$, and in t iterations the absolute value of any state coordinate $|s_t(l)|$ is at most T, the first summation is at most $(k-1)e^{2\eta T}$ and the second summation is at least $e^{-2\eta T}$. Therefore the previous expression is at most

$$\begin{aligned}
& (k-1)\left(e^{\eta}+e^{-\eta}-2\right)e^{2\eta T}-(e^{\eta}-1)e^{-2\eta T} \\
= \quad & (e^{\eta}-1)e^{-2\eta T}\left\{(k-1)(1-e^{-\eta})e^{4\eta T}-1\right\}.
\end{aligned}$$

We show that the term in curly brackets is negative. Firstly, using $e^x \geq 1+x$, we have $1-e^{-\eta} \leq \eta \leq 1/(4(k-1))$ by choice of η. Therefore it suffices to show that $e^{4\eta T}<4$. By choice of η again, $e^{4\eta T} \leq e^1 < 4$. This completes our proof. ■

The above lemma seems to suggest that under certain conditions, a sort of degeneracy occurs, and the optimal Booster payoff (15) is nearly unaffected by whether we use the minimal weak learning condition, or the condition $(\mathcal{C}^{\mathrm{eor}}, \mathbf{U}_\gamma)$. Indeed, we next prove this fact.

Theorem 20 *Suppose the loss function is as in Lemma 19, and for some parameter $\varepsilon > 0$, the number of examples m is large enough*

$$m \geq \frac{Te^{1/4}}{\varepsilon}. \tag{51}$$

Consider the minimal weak learning condition (13)*, and the fixed edge-over-random condition $(\mathcal{C}^{eor}, \mathbf{U}_\gamma)$ corresponding to the γ-biased uniform baseline $\mathbf{U}_\gamma$. Then the optimal booster payoffs using either condition is within ε of each other.*

Proof We show that the OS strategies arising out of either condition is the same. In other words, at any iteration t and state $\mathbf{s}_t$, both strategies play the same cost matrix and enforce the same constraints on the response of Weak-Learner. The theorem will then follow if we can invoke Theorems 9 and 16. For that, the number of examples needs to be as large as in (25). The required largeness would follow from (51) if the loss function satisfied (24) with $\varnothing(L,T)$ at most $\exp(1/4)$. Since the largest discrepancy in losses between two states reachable in T iterations is at most $e^{\eta T}-0$, the bound follows from the choice of η in (46). Therefore, it suffices to show the equivalence of the OS strategies corresponding to the two weak learning conditions.

We first show both strategies play the same cost-matrix. Lemma 19 states that the potential function using the minimal weak learning condition is the same as when using the fixed condition $(\mathcal{C}^{\mathrm{eor}}, \mathbf{U}_\gamma)$: $\phi_t = \phi_t^{\mathbf{u}}$, where $\mathbf{u}$ is as in (47). Since, according to (28) and (41), given a state $\mathbf{s}_t$ and iteration t, the two strategies choose cost matrices that are identical functions of the respective potentials, by the equivalence of the potential functions, the resulting cost matrices must be the same.

Even with the same cost matrix, the two different conditions could be imposing different constraints on Weak-Learner, which might affect the final payoff. For instance, with the baseline $\mathbf{U}_\gamma$, Weak-Learner has to return a weak classifier h satisfying

$$\mathbf{C}_t \bullet \mathbf{1}_h \leq \mathbf{C}_t \bullet \mathbf{U}_\gamma,$$

whereas with the minimal condition, the constraint on h is

$$\mathbf{C}_t \bullet \mathbf{1}_h \leq \max_{\mathbf{B} \in \mathcal{B}_\gamma^{\text{eor}}} \mathbf{C}_t \bullet \mathbf{B}.$$

In order to show that the constraints are the same we therefore need to show that for the common cost matrix $\mathbf{C}_t$ chosen, the right hand side of the two previous expressions are the same:

$$\mathbf{C}_t \bullet \mathbf{U}_\gamma = \max_{\mathbf{B} \in \mathcal{B}_\gamma^{\text{eor}}} \mathbf{C}_t \bullet \mathcal{B}_\gamma^{\text{eor}}. \tag{52}$$

We will in fact show the stronger fact that the equality holds for every row separately:

$$\forall i : \langle \mathbf{C}_t(i), \mathbf{u} \rangle = \max_{\mathbf{b} \in \Delta_\gamma^k} \langle \mathbf{C}_t(i), \mathbf{b} \rangle. \tag{53}$$

To see this, first observe that the choice of the optimal cost matrix $\mathbf{C}_t$ in (41) implies the following identity

$$\langle \mathbf{C}_t(i), \mathbf{b} \rangle = \mathbb{E}_{l \sim \mathbf{b}} \left[\phi_{T-t-1}(\mathbf{s}_t(i) + \mathbf{e}_l) \right].$$

On the other hand, (44) and Lemma 19 together imply that the distribution $\mathbf{b}$ maximizing the right hand side of the above is the γ-biased uniform distribution, from which (53) follows. Therefore, the constraints placed on Weak-Learner by the cost-matrix $\mathbf{C}_t$ is the same whether we use minimum weak learning condition or the fixed condition $(\mathcal{C}^{\text{eor}}, \mathbf{U}_\gamma)$. ■

One may wonder why η would be chosen so small, especially since the previous theorem indicates that such choices for η lead to degeneracies. To understand this, recall that η represents the size of the weights α_t chosen in every round, and was introduced as a tunable parameter to help achieve the best possible upper bound on zero-one error. More precisely, recall that the exponential loss $L_\eta^{\exp}(\mathbf{s})$ of the unweighted state, defined in (32), is equal to the exponential loss $L^{\exp}(\mathbf{f})$ on the weighted state, defined in (33), which in turn is an upper bound on the error $L^{\text{err}}(\mathbf{f}_T)$ of the final weighted state $\mathbf{f}_T$. Therefore the potential value $\phi_T(\mathbf{0})$ based on the exponential loss $L_\eta^{\exp}$ is an upper bound on the minimum error attainable after T rounds of boosting. At the same time, $\phi_T(\mathbf{0})$ is a function of η. Therefore, we may tune this parameter to attain the best bound possible. Even with this motivation, it may seem that a properly tuned η will not be as small as in Lemma 19, especially since it can be shown that the resulting loss bound $\phi_T(\mathbf{0})$ will always be larger than a fixed constant (depending on γ, k), no matter how many rounds T of boosting is used. However, the next result identifies conditions under which the tuned value of η is indeed as small as in Lemma 19. This happens when the edge γ is very small, as is described in the next theorem. Intuitively, a weak classifier achieving small edge has low accuracy, and a low weight reflects Booster's lack of confidence in this classifier.

Theorem 21 *When using the exponential loss function* (32)*, and the minimal weak learning condition* (13)*, the loss upper bound* $\phi_T(\mathbf{0})$ *provided by Theorem 16 is more than* 1 *and hence trivial unless the parameter* η *is chosen sufficiently small compared to the edge* γ*:*

$$\eta \leq \frac{k\gamma}{1-\gamma}. \tag{54}$$

In particular, when the edge is very small:

$$\gamma \leq \min\left\{\frac{1}{2}, \frac{1}{8k}\min\left\{\frac{1}{k}, \frac{1}{T}\right\}\right\}, \tag{55}$$

the value of η *needs to be as small as in* (46)*.*

Proof Comparing solutions (45) and (31) to the potentials corresponding to the minimal weak learning condition and a fixed edge-over-random condition, we may conclude that the loss bound $\phi_T(\mathbf{0})$ is in the former case is larger than $\phi_T^{\mathbf{b}}(\mathbf{0})$, for any edge-over-random distribution $\mathbf{b} \in \Delta_\gamma^k$. In particular, when $\mathbf{b}$ is set to be the γ-biased uniform distribution $\mathbf{u}$, as defined in (47), we get $\phi_T(\mathbf{0}) \geq \phi_T^{\mathbf{u}}(\mathbf{0})$. Now the latter bound, according to (34), is $\kappa(\gamma,\eta)^T$, where κ is defined as in (35). Therefore, to get non-trivial loss bounds which are at most 1, we need to choose η such that $\kappa(\gamma,\eta) \leq 1$. By (35), this happens when

$$\begin{aligned} \left(1-e^{-\eta}\right)\gamma &\geq \left(e^{\eta}+e^{-\eta}-2\right)\left(\frac{1-\gamma}{k}\right) \\ \text{i.e., } \frac{k\gamma}{1-\gamma} &\geq \frac{e^{\eta}+e^{-\eta}-2}{1-e^{-\eta}} = e^{\eta}-1 \geq \eta. \end{aligned}$$

Therefore (54) holds. When γ is as small as in (55), then $1-\gamma \leq \frac{1}{2}$, and therefore, by (54), the bound on η becomes identical to that in (55). ■

The condition in the previous theorem, that of the existence of only a very small edge, is the most we can assume for most practical data sets. Therefore, in such situations, we can compute the optimal Booster strategy that uses the minimal weak learning conditions. More importantly, using this result, we derive, in the next section, a highly efficient and practical *adaptive* algorithm, that is, one that does not require any prior knowledge about the edge γ, and will therefore work with any data set.

8. Variable Edges

So far we have required Weak-Learner to beat random by at least a fixed amount $\gamma > 0$ in each round of the boosting game. In reality, the edge over random is larger initially, and gets smaller as the OS algorithm creates harder cost matrices. Therefore requiring a fixed edge is either unduly pessimistic or overly optimistic. If the fixed edge is too small, not enough progress is made in the initial rounds, and if the edge is too large, Weak-Learner fails to meet the weak-learning condition in latter rounds. We fix this by not making any assumption about the edges, but instead *adaptively* responding to the edges returned by Weak-Learner. In the rest of the section we describe the adaptive procedure, and the resulting loss bounds guaranteed by it.

The philosophy behind the adaptive algorithm is a boosting game where Booster and Weak Learner no longer have opposite goals, but cooperate to reduce error as fast as possible. However, in order to create a clean abstraction and separate implementations of the boosting algorithms and the weak learning procedures as much as possible, we assume neither of the players has any knowledge of the details of the algorithm employed by the other player. In particular Booster may only assume that Weak Learner's strategy is barely strong enough to guarantee boosting. Therefore, Booster's demands on the weak classifiers returned by Weak Learner should be minimal, and it should send the weak learning algorithm the "easiest" cost matrices that will ensure boostability. In turn, Weak Learner may only assume a very weak Booster strategy, and therefore return a weak classifier that performs as well as possible with respect to the cost matrix sent by Booster.

At a high level, the adaptive strategy proceeds as follows. At any iteration, based on the states of the examples and number of remaining rounds of boosting, Booster chooses the game-theoretically optimal cost matrix assuming only infinitesimal edges in the remaining rounds. Intuitively, Booster has no high expectations of Weak Learner, and supplies it the easiest cost matrices with which it may be able to boost. However, in the adaptive setting, Weak-Learner is no longer adversarial. Therefore, although only infinitesimal edges are anticipated by Booster, Weak Learner cooperates in returning weak classifiers that achieve as large edges as possible, which will be more than just inifinitesimal. Based on the exact edge received in each round, Booster chooses the weight α_t adaptively to reach the most favourable state possible. Therefore, Booster plays game theoretically assuming an adversarial Weak Learner and expecting only the smallest edges in the future rounds, although Weak Learner actually cooperates, and Booster adaptively exploits this favorable behavior as much as possible. This way the boosting algorithm remains robust to a poorly performing Weak Learner, and yet can make use of a powerful weak learning algorithm whenever possible.

We next describe the details of the adaptive procedure. With variable weights we need to work with the weighted state $\mathbf{f}_t(i)$ of each example i, defined in (19). To keep the computations tractable, we will only be working with the exponential loss $L^{\exp}(\mathbf{f})$ on the weighted states. We first describe how Booster chooses the cost-matrix in each round. Following that we describe how it adaptively computes the weights in each round based on the edge of the weak classifier received.

8.1 Choosing the Cost-matrix

As discussed before, at any iteration t and state $\mathbf{f}_t$ Booster assumes that it will receive an infinitesimal edge γ in each of the remaining rounds. Since the step size is a function of the edge, which in turn is expected to be the same tiny value in each round, we may assume that the step size in each round will also be some fixed value η. We are therefore in the setting of Theorem 21, which states that the parameter η in the exponential loss function (32) should also be tiny to get any non-trivial bound. But then the loss function satisfies the conditions in Lemma 19, and by Theorem 20, the game theoretically optimal strategy remains the same whether we use the minimal condition or $(\mathcal{C}^{\text{eor}}, \mathbf{U}_\gamma)$. When using the latter condition, the optimal choice of the cost-matrix at iteration t and state $\mathbf{f}_t$, according to (36), is

$$C_t(i,l) = \begin{cases} (e^{\eta} - 1)\, e^{f_{t-1}(i,j) - f_{t-1}(i,1)} & \text{if } l > 1, \\ (e^{-\eta} - 1) \sum_{j=2}^{k} e^{f_{t-1}(i,j) - f_{t-1}(i,1)} & \text{if } l = 1. \end{cases}$$

Further, when using the condition $(\mathcal{C}^{\text{eor}}, \mathbf{U}_\gamma)$, the average potential of the states $\mathbf{f}_t(i)$, according to (34), is given by the average loss (37) of the state times $\kappa(\gamma,\eta)^{T-t}$, where the function κ is defined in (35). Our goal is to choose η as a function of γ so that $\kappa(\gamma,\eta)$ is as small as possible. Now, there is no lower bound on how small the edge γ may get, and, anticipating the worst, it makes sense to choose an infinitesimal γ, in the spirit of Freund (2001). Equation (35) then implies that the choice of η should also be infinitesimal. Then the above choice of the cost matrix becomes the following (after some rescaling):

$$\begin{aligned} C_t(i,l) &= \lim_{\eta\to 0} C_\eta(i,l) \triangleq \frac{1}{\eta}\begin{cases} (e^{\eta}-1)\, e^{f_{t-1}(i,j)-f_{t-1}(i,1)} & \text{if } l>1, \\ (e^{-\eta}-1)\sum_{j=2}^{k} e^{f_{t-1}(i,j)-f_{t-1}(i,1)} & \text{if } l=1. \end{cases} \\ &= \begin{cases} e^{f_{t-1}(i,j)-f_{t-1}(i,1)} & \text{if } l>1, \\ -\sum_{j=2}^{k} e^{f_{t-1}(i,j)-f_{t-1}(i,1)} & \text{if } l=1. \end{cases} \end{aligned} \tag{56}$$

We have therefore derived the optimal cost matrix played by the adaptive boosting strategy, and we record this fact.

Lemma 22 *Consider the boosting game using the minimal weak learning condition* (13). *Then, in iteration t at state* $\mathbf{f}_t$, *the game-theoretically optimal Booster strategy chooses the cost matrix* $\mathbf{C}_t$ *given in* (56).

We next show how to adaptively choose the weights α_t.

8.2 Adaptively Choosing Weights

Once Weak Learner returns a weak classifier h_t, Booster chooses the optimum weight α_t so that the resulting states $\mathbf{f}_t = \mathbf{f}_{t-1} + \alpha_t \mathbf{1}_{h_t}$ are as favorable as possible, that is, minimizes the total potential of its states. By our previous discussions, these are proportional to the total loss given by $Z_t = \sum_{i=1}^{m}\sum_{l=2}^{k} e^{f_t(i,l)-f_t(i,1)}$. For any choice of α_t, the difference $Z_t - Z_{t-1}$ between the total loss in rounds $t-1$ and t is given by

$$\begin{aligned} & (e^{\alpha_t}-1)\sum_{i\in S_-} e^{f_{t-1}(i,h_t(i))-f_{t-1}(i,1)} - \left(1-e^{-\alpha_t}\right)\sum_{i\in S_+} L^{\exp}(\mathbf{f}_{t-1}(i)) \\ = \;& (e^{\alpha_t}-1)A_-^t - \left(1-e^{-\alpha_t}\right)A_+^t \\ = \;& \left(A_+^t e^{-\alpha_t} + A_-^t e^{\alpha_t}\right) - \left(A_+^t + A_-^t\right), \end{aligned}$$

where S_+ denotes the set of examples that h_t classifies correctly, S_- the incorrectly classified examples, and A_-^t, A_+^t denote the first and second summations, respectively. Therefore, the task of choosing α_t can be cast as a simple optimization problem minimizing the previous expression. In fact, the optimal value of α_t is given by the following closed form expression

$$\alpha_t = \frac{1}{2}\ln\left(\frac{A_+^t}{A_-^t}\right). \tag{57}$$

With this choice of weight, one can show (with some straightforward algebra) that the total loss of the state falls by a factor less than 1. In fact the factor is exactly

$$(1-c_t) - \sqrt{c_t^2 - \delta_t^2}, \tag{58}$$

where

$$c_t = (A_+^t + A_-^t)/Z_{t-1},$$

and δ_t is the edge of the returned classifier h_t on the supplied cost-matrix $\mathbf{C}_t$. Notice that the quantity c_t is at most 1, and hence the factor (58) can be upper bounded by $\sqrt{1-\delta_t^2}$. We next show how to compute the edge δ_t. The definition of the edge depends on the weak learning condition being used, and in this case we are using the minimal condition (13). Therefore the edge δ_t is the largest γ such that the following still holds

$$\mathbf{C}_t \bullet \mathbf{1}_h \leq \max_{\mathbf{B} \in \mathcal{B}_\gamma^{\text{eor}}} \mathbf{C}_t \bullet \mathbf{B}.$$

However, since $\mathbf{C}_t$ is the optimal cost matrix when using exponential loss with a tiny value of η, we can use arguments in the proof of Theorem 20 to simplify the computation. In particular, eq. (52) implies that the edge δ_t may be computed as the largest γ satisfying the following simpler inequality

$$\begin{aligned}
\delta_t &= \sup\left\{\gamma : \mathbf{C}_t \bullet \mathbf{1}_{h_t} \leq \mathbf{C}_t \bullet \mathbf{U}_\gamma\right\} \\
&= \sup\left\{\gamma : \mathbf{C}_t \bullet \mathbf{1}_{h_t} \leq -\gamma \sum_{i=1}^{m}\sum_{l=2}^{k} e^{f_{t-1}(i,l)-f_{t-1}(i,1)}\right\} \\
\implies \delta_t &= \gamma : \ \mathbf{C}_t \bullet \mathbf{1}_{h_t} = -\gamma \sum_{i=1}^{m}\sum_{l=2}^{k} e^{f_{t-1}(i,l)-f_{t-1}(i,1)} \\
\implies \delta_t &= \frac{-\mathbf{C}_t \bullet \mathbf{1}_{h_t}}{\sum_{i=1}^{m}\sum_{l=2}^{k} e^{f_{t-1}(i,l)-f_{t-1}(i,1)}} = \frac{-\mathbf{C}_t \bullet \mathbf{1}_{h_t}}{Z_t},
\end{aligned} \tag{59}$$

where the first step follows by expanding $\mathbf{C}_t \bullet \mathbf{U}_\gamma$. We have therefore an adaptive strategy which efficiently reduces error. We record our results.

Lemma 23 *If the weight α_t in each round is chosen as in* (57)*, and the edge δ_t is given by* (59)*, then the total loss Z_t falls by the factor given in* (58)*, which is at most $\sqrt{1-\delta_t^2}$.*

The choice of α_t in (57) is optimal, but depends on quantities other than just the edge δ_t. We next show a way of choosing α_t based only on δ_t that still causes the total loss to drop by a factor of $\sqrt{1-\delta_t^2}$.

Lemma 24 *Suppose cost matrix $\mathbf{C}_t$ is chosen as in* (56)*, and the returned weak classifier h_t has edge δ_t, that is, $\mathbf{C}_t \bullet \mathbf{1}_{h_t} \leq \mathbf{C}_t \bullet \mathbf{U}_{\delta_t}$. Then choosing any weight $\alpha_t > 0$ for h_t makes the loss Z_t at most a factor*

$$1 - \frac{1}{2}(e^{\alpha_t} - e^{-\alpha_t})\delta_t + \frac{1}{2}(e^{\alpha_t} + e^{-\alpha_t} - 2)$$

of the previous loss Z_{t-1}. In particular by choosing

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1+\delta_t}{1-\delta_t}\right), \tag{60}$$

the drop factor is at most $\sqrt{1-\delta_t^2}$.

Proof We borrow notation from earlier discussions. The edge-condition implies

$$A_-^t - A_+^t = \mathbf{C}_t \bullet \mathbf{1}_{h_t} \leq \mathbf{C}_t \bullet \mathbf{U}_{\delta_t} = -\delta_t Z_{t-1} \implies A_+^t - A_-^t \geq \delta_t Z_{t-1}.$$

On the other hand, the drop in loss after choosing h_t with weight α_t is

$$\begin{aligned} & \left(1-e^{-\alpha_t}\right)A_+^t - (e^{\alpha_t}-1)A_-^t \\ = & \left(\frac{e^{\alpha_t}-e^{-\alpha_t}}{2}\right)\left(A_+^t - A_-^t\right) - \left(\frac{e^{\alpha_t}+e^{-\alpha_t}-2}{2}\right)\left(A_+^t + A_-^t\right). \end{aligned}$$

We have already shown that $A_+^t - A_-^t \geq \delta_t Z_{t-1}$. Further, $A_+^t + A_-^t$ is at most Z_{t-1}. Therefore the loss drops by a factor of at least

$$1-\frac{1}{2}(e^{\alpha_t}-e^{-\alpha_t})\delta_t + \frac{1}{2}(e^{\alpha_t}+e^{-\alpha_t}-2) = \frac{1}{2}\left\{(1-\delta_t)e^{\alpha_t} + (1+\delta_t)e^{-\alpha_t}\right\}.$$

Tuning α_t as in (60) causes the drop factor to be at least $\sqrt{1-\delta_t^2}$. ■

Algorithm 1 contains pseudocode for the adaptive algorithm, and includes both ways of choosing α_t. We call both versions of this algorithm AdaBoost.MM. With the approximate way of choosing the step length in (61), AdaBoost.MM turns out to be identical to AdaBoost.M2 (Freund and Schapire, 1997) or AdaBoost.MR (Schapire and Singer, 1999), provided the weak classifier space is transformed in an appropriate way to be acceptable by AdaBoost.M2 or AdaBoost.MR. We emphasize that AdaBoost.MM and AdaBoost.M2 are products of very different theoretical considerations, and this similarity should be viewed as a coincidence arising because of the particular choice of loss function, infinitesimal edge and approximate step size. For instance, when the step sizes are chosen instead as in (62), the training error falls more rapidly, and the resulting algorithm is different.

As a summary of all the discussions in the section, we record the following theorem.

Theorem 25 *The boosting algorithm AdaBoost.MM, shown in Algorithm 1, is the optimal strategy for playing the adaptive boosting game, and is based on the minimal weak learning condition. Further if the edges returned in each round are* $\delta_1,\ldots,\delta_T$, *then the error after* T *rounds is* $(k-1)\prod_{t=1}^T \sqrt{1-\delta_t^2} \leq (k-1)\exp\left\{-(1/2)\sum_{t=1}^T \delta_t^2\right\}$.

In particular, if a weak hypothesis space is used that satisfies the optimal weak learning condition (13)*, for some* γ*, then the edge in each round is large,* $\delta_t \geq \gamma$*, and therefore the error after* T *rounds is exponentially small,* $(k-1)e^{-T\gamma^2/2}$.

The theorem above states that as long as the minimal weak learning condition is satisfied, the error will decrease exponentially fast. Even if the condition is not satisfied, the error rate will keep falling rapidly provided the edges achieved by the weak classifiers are relatively high. However, our theory so far can provide no guarantees on these edges, and therefore it is not clear what is the best error rate achievable in this case, and how quickly it is achieved. The assumptions of boostability, and hence our minimal weak learning condition does not hold for the vast majority of practical data sets, and as such it is important to know what happens in such settings. In particular, an important requirement is *empirical consistency*, where we want that for any given weak classifier space, the algorithm converge, if allowed to run forever, to the weighted combination of classifiers that minimizes error on the training set. Another important criterion is *universal consistency*, which requires that the algorithm converge, when provided sufficient training data, to the classifier combination that minimizes error on the test data set. In the next section, we show that AdaBoost.MM satisfies such consistency requirements. Both the choice of the minimal weak learning condition as well as the setup of the adaptive game framework will play crucial roles in ensuring consistency. These results therefore provide evidence that game theoretic considerations can have strong statistical implications.

Algorithm 1 AdaBoost.MM

```
Require: Number of classes $k$, number of examples $m$.
Require: Training set $\{(x_1,y_1),\dots,(x_m,y_m)\}$ with $y_i \in \{1,\dots,k\}$ and $x_i \in X$.
  • Initialize $m \times k$ matrix $f_0(i,l) = 0$ for $i = 1,\dots,m$, and $l = 1,\dots,k$.
  for $t = 1$ to $T$ do
    • Choose cost matrix $\mathbf{C}_t$ as follows:
        $C_t(i,l) \quad = \quad \begin{cases} e^{f_{t-1}(i,l)-f_{t-1}(i,y_i)} & \text{if } l \neq y_i, \\ -\sum_{l \neq y_i} e^{f_{t-1}(i,j)-f_{t-1}(i,y_i)} & \text{if } l = 1. \end{cases}$
    • Receive weak classifier $h_t : X \to \{1,\dots,k\}$ from weak learning algorithm
    • Compute edge $\delta_t$ as follows:
        $\delta_t = \frac{-\sum_{i=1}^m C_t(i,h_t(x_i))}{\sum_{i=1}^m \sum_{l \neq y_i} e^{f_{t-1}(i,l)-f_{t-1}(i,y_i)}}$
    • Choose $\alpha_t$ either as
        $\alpha_t = \frac{1}{2} \ln\left(\frac{1+\delta_t}{1-\delta_t}\right), \qquad (61)$
      or, for a slightly bigger drop in the loss, as
        $\alpha_t = \frac{1}{2} \ln\left(\frac{\sum_{i:h_t(x_i)=y_i} \sum_{l \neq y_i} e^{f_{t-1}(i,l)-f_{t-1}(i,y_i)}}{\sum_{i:h_t(x_i)\neq y_i} e^{f_{t-1}(i,h_t(x_i))-f_{t-1}(i,y_i)}}\right) \qquad (62)$
    • Compute $\mathbf{f}_t$ as:
        $f_t(i,l) = f_{t-1}(i,l) + \alpha_t \mathbf{1}\left[h_t(x_i) = l\right].$
  end for
  • Output weighted combination of weak classifiers $F_T : X \times \{1,\dots,k\} \to \mathbb{R}$ defined as:
        $F_T(x,l) = \sum_{t=1}^T \alpha_t \mathbf{1}\left[h_t(x) = l\right]. \qquad (63)$
  • Based on $F_T$, output a classifier $H_T : X \to \{1,\dots,k\}$ that predicts as
        $H_T(x) = \operatorname*{argmax}_{l=1}^{k} F_T(x,l).$
```

9. Consistency of the Adaptive Algorithm

The goal in a classification task is to design a classifier that predicts with high accuracy on unobserved or test data. This is usually carried out by ensuring the classifier fits training data well without being overly complex. Assuming the training and test data are reasonably similar, one can show that the above procedure achieves high test accuracy, or is consistent. Here we work in a probabilistic setting that connects training and test data by assuming both consist of examples and labels drawn from a common, unknown distribution.

Consistency for multiclass classification in the probabilistic setting has been studied by Tewari and Bartlett (2007), who show that, unlike in the binary setting, many natural approaches fail to achieve consistency. In this section, we show that AdaBoost.MM described in the previous section avoids such pitfalls and enjoys various consistency results. We begin by laying down some standard assumptions and setting up some notation. Then we prove our first result showing that our algorithm minimizes a certain exponential loss function on the training data at a fast rate. Next, we build upon this result and improve along two fronts: firstly we change our metric from exponential loss to the more relevant classification error metric, and secondly we show fast convergence on not just training data, but also the test set. For the proofs, we heavily reuse existing machinery in the literature.

Throughout the rest of this section we consider the version of AdaBoost.MM that picks weights according to the approximate rule in (61). All our results most probably hold with the other rule for picking weights in (62) as well, but we did not verify that. These results hold without any boostability requirements on the space $\mathcal{H}$ of weak classifiers, and are therefore widely applicable in practice. While we do not assume any weak learning condition, we will require a fully cooperating Weak Learner. In particular, we will require that in each round Weak Learner picks the weak classifier suffering minimum cost with respect to the cost matrix provided by the boosting algorithm, or equivalently achieves the highest edge as defined in (59). Such assumptions are both necessary and standard in the literature, and are frequently met in practice.

In order to state our results, we will need to setup some notation. The space of examples will be denoted by $\mathcal{X}$, and the set of labels by $\mathcal{Y} = \{1,\ldots,k\}$. We also fix a finite weak classifier space $\mathcal{H}$ consisting of classifiers $h : \mathcal{X} \to \mathcal{Y}$. We will be interested in functions $F : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that assign a score to every example and label pair. Important examples of such functions are the weighted majority combinations (63) output by the adaptive algorithm. In general, any such combination of the weak classifiers in space $\mathcal{H}$ is specified by some weight function $\alpha : \mathcal{H} \to \mathbb{R}$; the resulting function is denoted by $F_\alpha : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, and satisfies:

$$F_\alpha(x,l) = \sum_{h\in\mathcal{H}} \alpha(h)\mathbf{1}\left[h(x) = l\right].$$

We will be interested in measuring the average exponential loss of such functions. To measure this, we introduce the $\widehat{\text{risk}}$ operator:

$$\widehat{\text{risk}}(F) \triangleq \frac{1}{m}\sum_{i=1}^{m}\sum_{l\neq y_i} e^{F(x_i,l)-F(x_i,y_i)}.$$

With this setup, we can now state our simplest consistency result, which ensures that the algorithm converges to a weighted combination of classifiers in the space $\mathcal{H}$ that achieves the minimum exponential loss over the training set at an efficient rate.

Lemma 26 *The $\widehat{\text{risk}}$ of the predictions F_T, as defined in* (63)*, converges to that of the optimal predictions of any combination of the weak classifiers in $\mathcal{H}$ at the rate $O(1/T)$:*

$$\widehat{\text{risk}}(F_T) - \inf_{\alpha:\mathcal{H}\to\mathbb{R}} \widehat{\text{risk}}(F_\alpha) \leq \frac{C}{T},$$

where C is a constant depending only on the data set.

A slightly stronger result would state that the average exponential loss when measured with respect to the *test set*, and not just the empirical set, also converges. The test set is generated by some target distribution D over example label pairs, and we introduce the risk_D operator to measure the exponential loss for any function $F : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ with respect to D:

$$\text{risk}_D(F) = \mathbb{E}_{(x,y)\sim D}\left[\sum_{l\neq y} e^{F(x,l)-F(x,y)}\right].$$

We show this stronger result holds if the function F_T is modified to the function $\bar{F}_T : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that takes values in the range $[0,-C]$, for some large constant C:

$$\bar{F}_T(x,l) \triangleq \max\left\{-C, F_T(x,l) - \max_{l'} F_T(x,l')\right\}. \tag{64}$$

Lemma 27 *If $\bar{F}_T$ is as in* (64)*, and the number of rounds T is set to $T_m = \sqrt{m}$, then its* risk_D *converges to the optimal value as $m \to \infty$ with high probability:*

$$\Pr\left[\text{risk}_D\left(\bar{F}_{T_m}\right) \leq \inf_{F:\mathcal{X}\times\mathcal{Y}\to\mathbb{R}} \text{risk}_D(F) + O\left(m^{-c}\right)\right] \geq 1 - \frac{1}{m^2},$$

where $c > 0$ is some absolute constant, and the probability is over the draw of training examples.

We prove Lemmas 26 and 27 by demonstrating a strong correspondence between AdaBoost.MM and binary AdaBoost, and then leveraging almost identical known consistency results for AdaBoost (Bartlett and Traskin, 2007). Our proofs will closely follow the exposition in Chapter 12 of Schapire and Freund (2012) on the consistency of AdaBoost, and are deferred to the appendix.

So far we have focused on risk_D, but a more desirable consistency result would state that the test *error* of the final classifier output by AdaBoost.MM converges to the Bayes optimal error. The test error is measured by the err_D operator, and is given by

$$\text{err}_D(H) = \Pr_{(x,y)\sim D}\left[H(x) \neq y\right].$$

The Bayes optimal classifier H_{opt} is a classifier achieving the minimum error among all possible classifying functions

$$\text{err}_D(H_{\text{opt}}) = \inf_{H:\mathcal{X}\to\mathcal{Y}} \text{err}_D(H),$$

and we want our algorithm to output a classifier whose err_D approaches $\text{err}_D(H_{\text{opt}})$. In designing the algorithm, our main focus was on reducing the exponential loss, captured by risk_D and $\widehat{\text{risk}}$. Unless these loss functions are aligned properly with classification error, we cannot hope to achieve optimal error. The next result shows that our loss functions are correctly aligned, or more technically *Bayes consistent*. In other words, if a scoring function $F : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is close to achieving optimal risk_D, then the classifier $H : \mathcal{X} \to \mathcal{Y}$ derived from it as follows:

$$H(x) \in \operatorname*{argmax}_{l\in\mathcal{Y}} F(x,y), \tag{65}$$

also approaches the Bayes optimal error.

Lemma 28 *Suppose F is a scoring function achieving close to optimal risk*

$$\text{risk}_D(F) \leq \inf_{F' : \mathcal{X}\times\mathcal{Y}\to\mathbb{R}} \text{risk}_D(F') + \varepsilon, \tag{66}$$

for some $\varepsilon \geq 0$. If H is the classifier derived from it as in (65)*, then it achieves close to the Bayes optimal error*

$$\text{err}_D(H) \leq \text{err}_D(H_{\text{opt}}) + \sqrt{2\varepsilon}.$$

Proof The proof is similar to that of Theorem 12.1 in Schapire and Freund (2012), which in turn is based on the work by Zhang (2004) and Bartlett et al. (2006). Let $p(x) = \Pr_{(x',y')\sim D}(x' = x)$ denote the the marginalized probability of drawing example x from D, and let $p^x_y = \Pr_{(x',y')\sim D}[y' = y | x' = x]$ denote the conditional probability of drawing label y given we have drawn example x. We first rewrite the difference in errors between H and H_{opt} using these probabilities. Firstly note that the accuracy of any classifier H' is given by

$$\sum_{x\in\mathcal{X}} D(x, H'(x)) = \sum_{x\in\mathcal{X}} p(x) p^x_{H'(x)}.$$

If $\mathcal{X}'$ is the set of examples where the predictions of H and H_{opt} differ, $\mathcal{X}' = \left\{x \in \mathcal{X} : H(x) \neq H_{\text{opt}}(x)\right\}$, then we may bound the error differences as

$$\text{err}_D(H) - \text{err}_D(H_{\text{opt}}) = \sum_{x\in\mathcal{X}'} p(x)\left(p^x_{H_{\text{opt}}(x)} - p^x_{H(x)}\right). \tag{67}$$

We next relate this expression to the difference of the losses.

Notice that for any scoring function F', the risk$_D$ can be rewritten as follows :

$$\text{risk}_D(F') = \sum_{x\in\mathcal{X}} p(x) \sum_{l<l'} \left\{ p^x_l e^{F'(x,l') - F'(x,l)} + p^x_{l'} e^{F'(x,l) - F'(x,l')} \right\}.$$

Denote the inner summation in curly brackets by $L^{l,l'}_{F'}(x)$, and notice this quantity is minimized if

$$e^{F'(x,l) - F'(x,l')} = \sqrt{p^x_l / p^x_{l'}}, \quad \text{i.e., if } F'(x,l) - F'(x,l') = \tfrac{1}{2}\ln p^x_l - \tfrac{1}{2}\ln p^x_{l'}.$$

Therefore, defining $F^*(x,l) = \frac{1}{2}\ln p^x_l$ leads to a risk$_D$ minimizing function F^*. Furthermore, for any example and pair of labels l, l', the quantity $L^{l,l'}_{F^*}(x)$ is at most $L^{l,l'}_F(x)$, and therefore the difference in losses of F^* and F may be lower bounded as follows:

$$\begin{aligned} \varepsilon \geq \text{risk}_D(F) - \text{risk}_D(F^*) &= \sum_{x\in\mathcal{X}} p(x) \sum_{l\neq l'} \left(L^{l,l'}_F - L^{l,l'}_{F^*}\right) \\ &\geq \sum_{x\in\mathcal{X}'} p(x) \left\{ L^{H(x),H_{\text{opt}}(x)}_F - L^{H(x),H_{\text{opt}}(x)}_{F^*} \right\}. \end{aligned} \tag{68}$$

We next study the term in the curly brackets for a fixed x. Let A and B denote $H(x)$ and $H_{\text{opt}}(x)$, respectively. We have already seen that $L^{A,B}_{F^*} = 2\sqrt{p^x_A p^x_B}$. Further, by definition of Bayes optimality,

$p_A^x \geq p_B^x$. On the other hand, since $x \in \mathcal{X}'$, we know that $B \neq A$, and hence, $F(x,A) \geq F(x,B)$. Let $e^{F(x,B)-F(x,A)} = 1+\eta$, for some $\eta \geq 0$. The quantity $L_F^{A,B}$ may be lower bounded as:

$$\begin{aligned} L_F^{A,B} &= p_A^x e^{F(x,B)-F(x,A)} + p_B^x e^{F(x,A)-F(x,B)} \\ &= (1+\eta)p_A^x + (1+\eta)^{-1} p_B^x \\ &\geq (1+\eta)p_A^x + (1-\eta)p_B^x \\ &= p_A^x + p_B^x + \eta(p_A^x - p_B^x) \geq p_A^x + p_B^x. \end{aligned}$$

Combining we get

$$L_F^{A,B} - L_{F^*}^{A,B} \geq p_A^x + p_B^x - 2\sqrt{p_A^x p_B^x} = \left(\sqrt{p_A^x} - \sqrt{p_B^x}\right)^2.$$

Plugging back into (68) we get

$$\sum_{x \in \mathcal{X}'} p(x) \left(\sqrt{p_{H(x)}^x} - \sqrt{p_{H_{\text{opt}}(x)}^x}\right)^2 \leq \varepsilon. \tag{69}$$

Now we connect (67) to the previous expression as follows

$$\begin{aligned} &\left\{\text{err}_D(H) - \text{err}_D(H_{\text{opt}})\right\}^2 \\ &= \left\{\sum_{x \in \mathcal{X}'} p(x) \left(p_{H_{\text{opt}}(x)}^x - p_{H(x)}^x\right)\right\}^2 \\ &\leq \left(\sum_{x \in \mathcal{X}'} p(x)\right) \left(\sum_{x \in \mathcal{X}'} p(x) \left(p_{H_{\text{opt}}(x)}^x - p_{H(x)}^x\right)^2\right) \text{ (Cauchy-Schwartz)} \\ &\leq \sum_{x \in \mathcal{X}'} p(x) \left(\sqrt{p_{H_{\text{opt}}(x)}^x} - \sqrt{p_{H(x)}^x}\right)^2 \left(\sqrt{p_{H_{\text{opt}}(x)}^x} + \sqrt{p_{H(x)}^x}\right)^2 \qquad (70) \\ &\leq 2\sum_{x \in \mathcal{X}'} p(x) \left(\sqrt{p_{H_{\text{opt}}(x)}^x} - \sqrt{p_{H(x)}^x}\right)^2 \qquad (71) \\ &\leq 2\varepsilon, \text{ (by (69))} \end{aligned}$$

where (70) holds since

$$\sum_{x \in \mathcal{X}'} p(x) = \Pr_{(x',y') \sim D}\left[x' \in \mathcal{X}'\right] \leq 1,$$

and (71) holds since

$$\begin{aligned} p_{H(x)}^x + p_{H_{\text{opt}}(x)}^x &= \Pr_{(x',y') \sim D}\left[y' \in \left\{H(x), H_{\text{opt}}(x)\right\} | x\right] \leq 1 \\ \implies \sqrt{p_{H(x)}^x} + \sqrt{p_{H_{\text{opt}}(x)}^x} &\leq \sqrt{2}. \end{aligned}$$

Therefore, $\text{err}_D(H) - \text{err}_D(H_{\text{opt}}) \leq \sqrt{2\varepsilon}$. ■

Note that the classifier $\bar{H}_T$, derived from the truncated scoring function $\bar{F}_T$ in the manner provided in (65), makes identical predictions to, and hence has the same err_D as, the classifier H_T output by the adaptive algorithm. Further, Lemma 27 seems to suggest that $\bar{F}_T$ satisfies the condition in (66), which, combined with our previous observation $\text{err}_D(H) = \text{err}_D(\bar{H}_T)$, would imply H_T approaches the optimal error. However, the condition (66) requires achieving optimal risk over all scoring functions, and not just ones achievable as a combination of weak classifiers in $\mathcal{H}$. Therefore, in order to use Lemma 28, we require the weak classifier space to be sufficiently rich, so that some combination of the weak classifiers in $\mathcal{H}$ attains risk_D arbitrarily close to the minimum attainable by any function:

$$\inf_{\alpha:\mathcal{H}\to\mathbb{R}} \text{risk}_D(F_\alpha) = \inf_{F:\mathcal{X}\times\mathcal{Y}\to\mathbb{R}} \text{risk}_D(F). \tag{72}$$

The richness condition, along with our previous arguments and Lemma 27, immediately imply the following result.

Theorem 29 *If the weak classifier space $\mathcal{H}$ satisfies the richness condition* (72)*, and the number of rounds T is set to $\sqrt{m}$, then the error of the final classifier H_T approaches the Bayes optimal error:*

$$\Pr\left[\text{err}_D\left(H_{\sqrt{m}}\right) \le \text{err}_D(H_{\text{opt}}) + O\left(m^{-c}\right)\right] \ge 1 - \frac{1}{m^2},$$

where $c > 0$ is some positive constant, and the probability is over the draw of training examples.

A consequence of the theorem is our strongest consistency result:

Corollary 30 *Let H_{opt} be the Bayes optimal classifier, and let the weak classifier space $\mathcal{H}$ satisfy the richness condition* (72)*. Suppose m example and label pairs $\{(x_1,y_1),\ldots,(x_m,y_m)\}$ are sampled from the distribution D, the number of rounds T is set to be $\sqrt{m}$, and these are supplied to AdaBoost.MM. Then, in the limit $m \to \infty$, the final classifier $H_{\sqrt{m}}$ output by AdaBoost.MM achieves the Bayes optimal error almost surely:*

$$\Pr\left[\left\{\lim_{m\to\infty} \text{err}_D(H_{\sqrt{m}})\right\} = \text{err}_D(H_{\text{opt}})\right] = 1,$$

where the probability is over the randomness due to the draw of training examples.

The proof of Corollary 30, based on the Borel-Cantelli Lemma, is very similar to that of Corollary 12.3 in Schapire and Freund (2012), and so we omit it. When $k = 2$, AdaBoost.MM is identical to AdaBoost. For Theorem 29 to hold for AdaBoost, the richness assumption (72) is necessary, since there are examples due to Long and Servedio (2010) showing that the theorem may not hold when that assumption is violated.

Although we have seen that technically AdaBoost.MM is consistent under broad assumptions, intuitively perhaps it is not clear what properties were responsible for this desirable behavior. We next briefly study the high level ingredients necessary for consistency in boosting algorithms.

9.1 Key Ingredients for Consistency

We show here how both the choice of the loss function as well as the weak learning condition play crucial roles in ensuring consistency. If the loss function were not Bayes consistent as in Lemma 28, driving it down arbitrarily could still lead to high test error. For example, the loss employed by SAMME (Zhu et al., 2009) does not upper bound the error, and therefore although it can manage to drive down its loss arbitrarily when supplied by the data set discussed in Figure 1, although its error remains high.

Equally important is the weak learning condition. Even if the loss function is chosen to be error, so that it is trivially Bayes consistent, choosing the wrong weak learning condition could lead to inconsistency. In particular, if the weak learning condition is stronger than necessary, then, even on a boostable data set where the error can be driven to zero, the boosting algorithm may get stuck prematurely because its stronger than necessary demands cannot be met by the weak classifier space. We have already seen theoretical examples of such data sets, and we will see some practical instances of this phenomenon in the next section.

On the other hand, if the weak learning condition is too weak, then a lazy Weak Learner may satisfy the Booster's demands by returning weak classifiers belonging only to a non-boostable subset of the available weak classifier space. For instance, consider again the data set in Figure 1, and assume that this time the weak classifier space is much richer, and consists of all possible classifying functions. However, in any round, Weak Learner searches through the space, first trying hypotheses h_1 and h_2 shown in the figure, and only if neither satisfy the Booster, search for additional weak classifiers. In that case, any algorithm using SAMME's weak learning condition, which is known to be too weak and satisfiable by just the two hypotheses $\{h_1, h_2\}$, would only receive h_1 or h_2 in each round, and therefore be unable to reach the optimum accuracy. Of course, if the Weak Learner is extremely generous and helpful, then it may return the right collection of weak classifiers even with a null weak learning condition that places no demands on it. However, in practice, many Weak Learners used are similar to the lazy weak learner described since these are computationally efficient.

To see the effect of inconsistency arising from too weak learning conditions in practice, we need to test boosting algorithms relying on such data sets on significantly hard data sets, where only the strictest Booster strategy can extract the necessary service from Weak Learner for creating an optimal classifier. We did not include such experiments, and it will be an interesting empirical conjecture to be tested in the future. However, we did include experiments that illustrate the consequence of using too strong conditions, and we discuss those in the next section.

10. Experiments

In the final section of this paper, we report preliminary experimental results on 13 UCI data sets: letter, nursery, pendigits, satimage, segmentation, vowel, car, chess, connect4, forest, magic04, poker, abalone. These data sets are all multiclass except for magic04, have a wide range of sizes, contain all combinations of real and categorical features, have different number of examples to number of features per example ratios, and are drawn from a variety of real-life situations. A summary of each data set is provided in Figure 7. Most sets come with prespecified train and test splits which we use; if not, we picked a random 4 : 1 split. Sometimes the prespecified test set was too large compared to the training set, and we restricted ourselves to the first ten thousand examples of the specified test set. Throughout this section by MM we refer to the version of AdaBoost.MM studied in the consistency section, which uses the approximate step size (61).

There were two kinds of experiments. In the first, we took a standard implementation M1 of AdaBoost.M1 with C4.5 as weak learner, and the Boostexter implementation MH of AdaBoost.MH using stumps (Schapire and Singer, 2000), and compared it against our method MM with a naive greedy tree-searching weak-learner Greedy. We will refer to the number of leaves as the *size* of

data set	classes	test	train	discrete	real
abalone*	28	1044	3133	1	7
car	4	345	1383	6	0
chess	2	799	2397	36	0
connect4	3	13511	54046	42	0
forest*	7	10000*	15120	44	10
letter	26	4000	16000	0	16
magic04	2	3804	15216	0	10
nursery	5	2591	10369	8	0
pendigits*	10	3498	7494	0	16
poker*	10	10000*	25010	5	5
satimage*	6	2000	4435	0	36
segmentation*	7	2100	210	0	19
vowel*	11	462	528	0	10

Figure 7: Summaries of the data sets used in the experiments. Each row contains the name of data set, number of labels, number of test examples, and training examples, and number of discrete and real features, in that order. The data sets that come with prespecified training-test splits are marked with an asterisk. When the prespecified test set was too large compared to the training set, only the first ten thousand examples were used. These test set sizes are marked with an asterisk.

the tree. Note that since the trees are full binary trees, with each internal node having exactly two children, the total number of nodes in the tree is one less than twice the number of leaves. When C4.5 is run as the weak learner, it grows the tree till a desired accuracy is reached in each round, and thereby automatically picks the tree sizes. Those sizes were used to pick the maximum size of the trees that Greedy was allowed to return when run as a weak learner by MM, so that both M1 and MM output ensembles of trees of roughly similar sizes. The test-errors after 500 rounds of boosting for each algorithm and data set are shown in Figure 8. The performance is comparable with M1 and far better than MH (understandably since stumps are far weaker than trees), even though our weak-learner is very naive. The convergence rates of error with rounds of M1 and MM are also comparable, as shown in Figure 9 (we omitted the curve for MH since it lay far above both M1 and MM).

We next investigated how each algorithm performs with less powerful weak-learners. We modified MH so that it uses a tree returning a single multiclass prediction on each example. For MH and MM we used the Greedy weak learner, while for M1 we used a more powerful-variant Greedy-Info whose greedy criterion was information gain rather than error or cost (we also ran M1 on top of Greedy but Greedy-Info consistently gave better results so we only report the latter). The reason for using weak learners that optimize different cost functions with the different boosting algorithms is as follows. M1 is based on the error-metric, where every example incurs a penalty of 0 when classified correctly and 1 when classified incorrectly. Information gain, measuring how “impure” the nodes resulting from a particular split are, is well aligned with the error metric. However, MH and MM use more general cost functions, and we could not come up with appropriate generalizations of information gain for these setting. So we just used the cost itself in deciding how to grow the de-

data set	MH	M1	MM
abalone	0.732	0.751	0.750
car	0.264	0.336	0.159
chess	0.025	0.003	0.005
connect4	0.321	0.306	0.282
forest	0.326	0.238	0.239
letter	0.146	0.031	0.027
magic04	0.148	0.117	0.118
nursery	0.081	0.164	0.196
pendigits	0.046	0.026	0.095
poker	0.497	0.341	0.228
satimage	0.121	0.088	0.093
segmentation	0.050	0.053	0.149
vowel	0.569	0.485	0.567

Figure 8: This is a table of the final test-errors of standard implementations of MH, M1 and MM after 500 rounds of boosting on different data sets. Both M1 and MM achieve comparable error, which is often smaller than that achieved by MH. This is because M1 and MM used trees of comparable sizes which were often much larger and powerful than the decision stumps that MH boosted.

cision tree. We tried all tree-sizes in the set $\{10, 20, 50, 100, 200, 500, 1000, 2000, 4000\}$ up to the tree-size used by M1 on C4.5 for each data-set. We plotted the error of each algorithm against tree size for each data-set in Figure 10. As predicted by our theory, our algorithm succeeds in boosting the accuracy even when the tree size is too small to meet the stronger weak learning assumptions of the other algorithms. More insight is provided by plots in Figure 11 of the rate of convergence of error with rounds when the tree size allowed is very small (5). Both M1 and MH drive down the error for a few rounds. But since boosting keeps creating harder distributions, very soon the small-tree learning algorithms Greedy and Greedy-Info are no longer able to meet the excessive requirements of M1 and MH respectively. However, our algorithm makes more reasonable demands that are easily met by Greedy.

11. Conclusion and Discussion

In summary, we create a new framework for studying multiclass boosting. This framework is very general and captures the weak learning conditions implicitly used by many earlier multiclass boosting algorithms as well as novel conditions, including the minimal condition under which boosting is possible. We also show how to design boosting algorithms relying on these weak learning conditions that drive down training error rapidly. These algorithms are the optimal strategies for playing certain two player games. Based on this game-theoretic approach, we also design a multiclass boosting algorithm that is consistent, that is, approaches the minimum empirical risk, and under some basic assumptions, the Bayes optimal test error. Preliminary experiments show that this algorithm can achieve much lower error compared to existing algorithms when used with very weak classifiers.

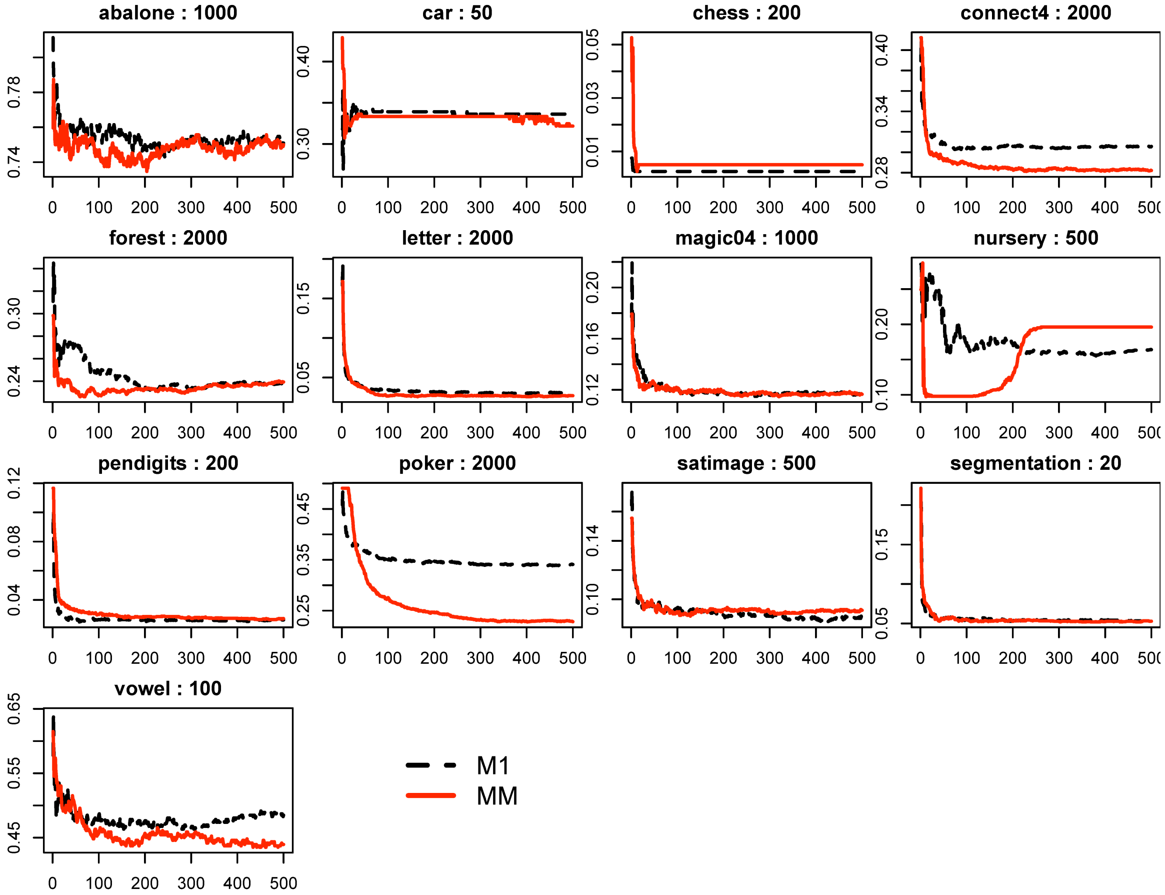


Figure 9: Plots of the rates at which `M1`(black,dashed) and `MM`(red,solid) drive down test-error on different data-sets when using trees of comparable sizes as weak classifiers. `M1` called C4.5, and `MM` called `Greedy`, respectively, as weak-learner. The tree sizes returned by C4.5 were used as a bound on the size of the trees that `Greedy` was allowed to return. This bound on the tree-size depended on the data set, and are shown next to the data set labels.

The notion of game-theoretic equivalence in Section 7.1 is based upon a weak learner that may return any weak hypothesis, which is absurd from a practical viewpoint. However, designing optimal boosting algorithms separately for different kinds of weak learners, which we leave as an open problem, will lead to a much more complex theory. Further, it is not clear what the additional gain (in terms of improvement in loss bounds) may be. Our philosophy here was to take the ultra-conservative approach, so that the resulting boosting algorithm enjoys bounds that hold under all settings. Theorem 14 then says, that in that ultra-conservative framework, the best algorithm remains the same if you change the weak-learning condition to another ”equivalent” condition.

Although we can efficiently compute the game-theoretically optimal strategies under most conditions, when using the minimal weak learning condition, and non-convex 0-1 error as loss function,

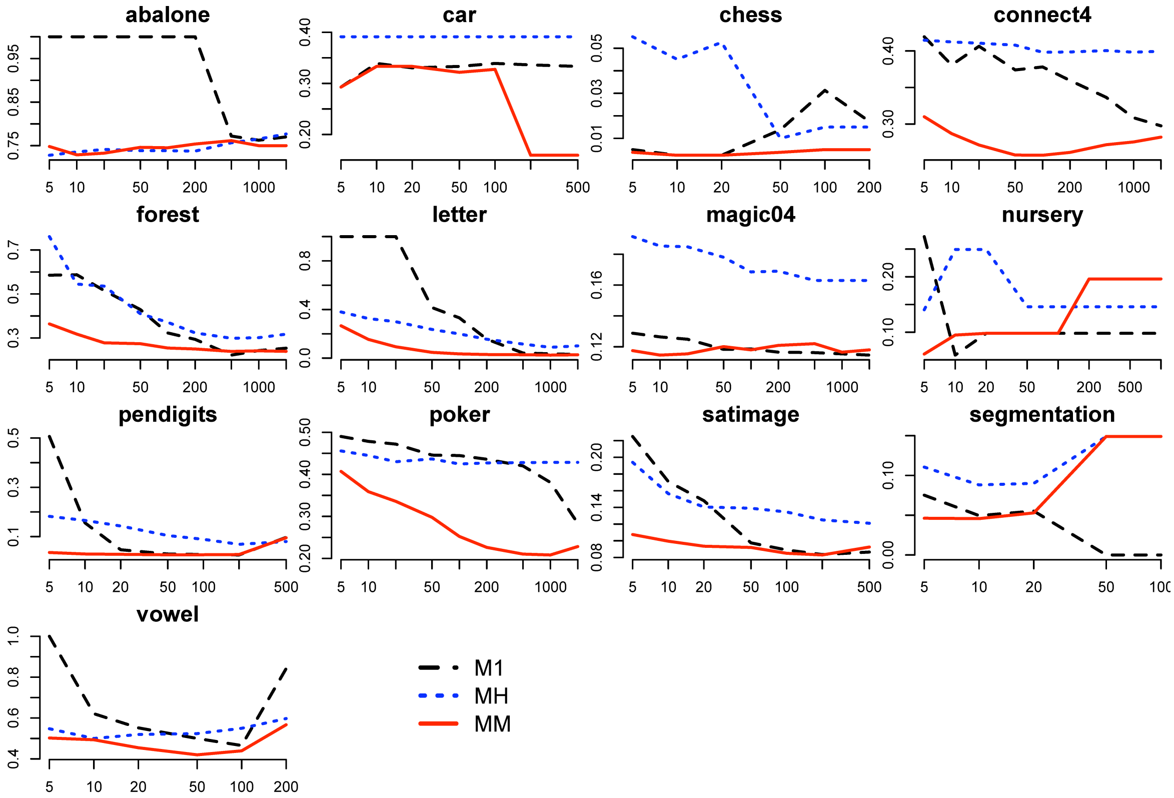


Figure 10: For this figure, M1(black, dashed), MH(blue, dotted) and MM(red,solid) were designed to boost decision trees of restricted sizes. The final test-errors of the three algorithms after 500 rounds of boosting are plotted against the maximum tree-sizes allowed for the weak classifiers. MM achieves much lower error when the weak classifiers are very weak, that is, with smaller trees.

we require exponential computational time to solve the corresponding boosting games. Boosting algorithms based on error are potentially far more noise tolerant than those based on convex loss functions, and finding efficiently computable near-optimal strategies in this situation is an important problem left for future work. Further, we primarily work with weak classifiers that output a single multiclass prediction per example, whereas weak hypotheses that make multilabel multiclass predictions are typically more powerful. We believe that multilabel predictions do not increase the power of the weak learner in our framework, and our theory can be extended without much work to include such hypotheses, but we do not address this here. Finally, it will be interesting to see if the notion of minimal weak learning condition can be extended to boosting settings beyond classification, such as ranking.

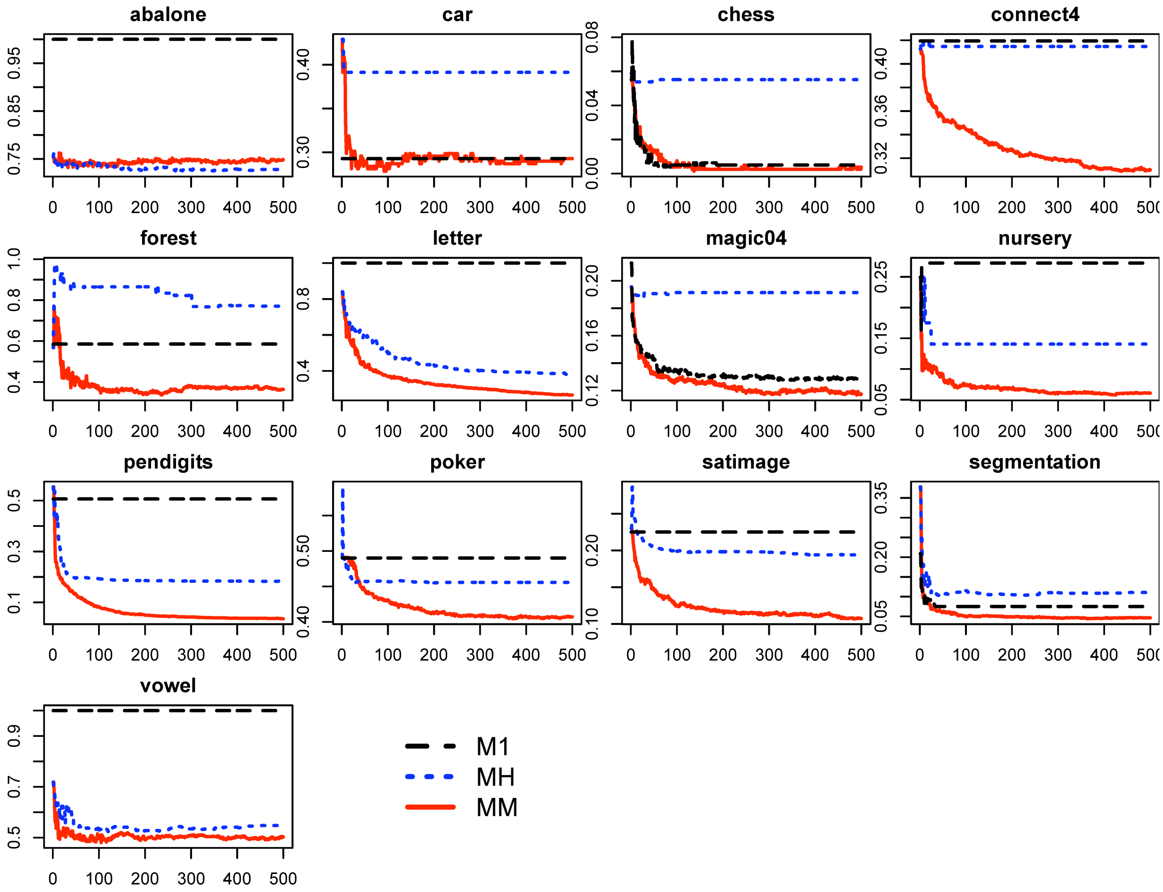


Figure 11: A plot of how fast the test-errors of the three algorithms drop with rounds when the weak classifiers are trees with a size of at most 5. Algorithms M1 and MH make strong demands which cannot be met by the extremely weak classifiers after a few rounds, whereas MM makes gentler demands, and is hence able to drive down error through all the rounds of boosting.

Acknowledgments

This research was funded by the National Science Foundation under grants IIS-0325500 and IIS-1016029.

Appendix A. Omitted Proofs

We include proofs and details that were omitted earlier in the paper.

A.1 Optimality of the OS Strategy

Here we prove Theorem 9. The proof of the upper bound on the loss is very similar to the proof of Theorem 2 in Schapire (2001). For the lower bound, a similar result is proved in Theorem 3 in Schapire (2001). However, the proof relies on certain assumptions that may not hold in our setting, and we instead follow the more direct lower bounding techniques in Section 5 of Mukherjee and Schapire (2010).

We first show that the average potential of states does not increase in any round. The dual form of the recurrence (21) and the choice of the cost matrix $\mathbf{C}_t$ in (22) together ensure that for each example i,

$$\begin{aligned}\phi^{\mathbf{B}(i)}_{T-t}(\mathbf{s}_t(i)) &= \max_{l=1}^{k}\left\{\phi^{\mathbf{B}(i)}_{T-t-1}(\mathbf{s}_t(i)+\mathbf{e}_l) - (\mathbf{C}_t(i)(l) - \langle \mathbf{C}_t(i), \mathbf{B}(i)\rangle)\right\}\\ &\geq \phi^{\mathbf{B}(i)}_{T-t-1}\left(\mathbf{s}_t(i)+\mathbf{e}_{h_t(x_i)}\right) - (C_t(i,h_t(x_i)) - \langle \mathbf{C}_t(i), \mathbf{B}(i)\rangle).\end{aligned}$$

Summing up the inequalities over all examples, we get

$$\sum_{i=1}^{m}\phi^{\mathbf{B}(i)}_{T-t-1}\left(\mathbf{s}_t(i)+\mathbf{e}_{h_t(x_i)}\right) \leq \sum_{i=1}^{m}\phi^{\mathbf{B}(i)}_{T-t}(\mathbf{s}_t(i)) + \sum_{i=1}^{m}\left\{C_t(i,h_t(x_i)) - \langle \mathbf{C}_t(i), \mathbf{B}(i)\rangle\right\}$$

The first two summations are the total potentials in round $t+1$ and t, respectively, and the third summation is the difference in the costs incurred by the weak-classifier h_t returned in iteration t and the baseline $\mathbf{B}$. By the weak learning condition, this difference is non-positive, implying that the average potential does not increase.

Next we show that the bound is tight. In particular choose any accuracy parameter $\varepsilon > 0$, and total number of iterations T, and let m be as large as in (25). We show that in any iteration $t \leq T$, based on Booster's choice of cost-matrix $\mathbf{C}$, an adversary can choose a weak classifier $h_t \in \mathcal{H}^{\text{all}}$ such that the weak learning condition is satisfied, and the average potential does not fall by more than an amount ε/T. In fact, we show how to choose labels $l_1, \ldots, l_m$ such that the following hold simultaneously:

$$\sum_{i=1}^{m} C(i,l_i) \leq \sum_{i=1}^{m}\langle \mathbf{C}(i), \mathbf{B}(i)\rangle \tag{73}$$

$$\sum_{i=1}^{m}\phi^{\mathbf{B}(i)}_{T-t}(\mathbf{s}_t(i)) \leq \frac{m\varepsilon}{T} + \sum_{i=1}^{m}\phi^{\mathbf{B}(i)}_{T-t-1}(\mathbf{s}_t(i)+\mathbf{e}_{l_i}) \tag{74}$$

This will imply that the final potential or loss is at least ε less than the bound in (23).

We first construct, for each example i, a distribution $\mathbf{p}_i \in \Delta\{1,\ldots,k\}$ such that the size of the support of $\mathbf{p}_i$ is either 1 or 2, and

$$\phi^{\mathbf{B}(i)}_{T-t}(\mathbf{s}_t(i)) = \mathbb{E}_{l\sim\mathbf{p}_i}\left[\phi^{\mathbf{B}(i)}_{T-t-1}(\mathbf{s}_t(i)+\mathbf{e}_l)\right]. \tag{75}$$

To satisfy (75), by (17), we may choose $\mathbf{p}_i$ as any optimal response of the max player in the minimax recurrence when the min player chooses $\mathbf{C}(i)$:

$$\begin{aligned}\mathbf{p}_i &\in \operatorname*{argmax}_{\mathbf{p}\in\mathcal{P}_i}\left\{\mathbb{E}_{l\sim\mathbf{p}}\left[\phi^{\mathbf{B}(i)}_{t-1}(\mathbf{s}+\mathbf{e}_l)\right]\right\} &(76)\\ \text{where } \mathcal{P}_i &= \left\{\mathbf{p}\in\Delta\{1,\ldots,k\} : \mathbb{E}_{l\sim\mathbf{p}}[C(i,l)] \le \langle \mathbf{C}(i),\mathbf{B}(i)\rangle\right\}. &(77)\end{aligned}$$

The existence of $\mathbf{p}_i$ is guaranteed, since, by Lemma 7, the polytope $\mathcal{P}_i$ is non-empty for each i. The next result shows that we may choose $\mathbf{p}_i$ to have a support of size 1 or 2.

Lemma 31 *There is a* $\mathbf{p}_i$ *satisfying* (76) *with either 1 or 2 non-zero coordinates.*

Proof Let $\mathbf{p}^*$ satisfy (76), and let its support set be S. Let μ_i denote the mean cost under this distribution:

$$\mu_i = \mathbb{E}_{l\sim\mathbf{p}^*}[C(i,l)] \le \langle \mathbf{C}(i),\mathbf{B}(i)\rangle.$$

If the support has size at most 2, then we are done. Further, if each non-zero coordinate $l\in S$ of $\mathbf{p}^*$ satisfies $C(i,l)=\mu_i$, then the distribution $\mathbf{p}_i$ that concentrates all its weight on the label $l^{\min}\in S$ minimizing $\phi^{\mathbf{B}(i)}_{t-1}(\mathbf{s}+\mathbf{e}_{l^{\min}})$ is an optimum solution with support of size 1. Otherwise, we can pick labels $l_1^{\min}, l_2^{\min}\in S$ such that

$$C(i,l_1^{\min}) < \mu_i < C(i,l_2^{\min}).$$

Then we may choose a distribution $\mathbf{q}$ supported on these two labels with mean μ_i:

$$\mathbb{E}_{l\sim\mathbf{q}}[C(i,l)] = q(l_1^{\min})C(i,l_1^{\min}) + q(l_2^{\min})C(i,l_2^{\min}) = \mu_i.$$

Choose λ as follows:

$$\lambda = \min\left\{\frac{p^*(l_1^{\min})}{q(l_1^{\min})}, \frac{p^*(l_2^{\min})}{q(l_2^{\min})}\right\},$$

and write $\mathbf{p}^* = \lambda\mathbf{q}+(1-\lambda)\mathbf{p}$. Then both $\mathbf{p},\mathbf{q}$ belong to the polytope $\mathcal{P}_i$, and have strictly fewer non-zero coordinates than $\mathbf{p}^*$. Further, by linearity, one of $\mathbf{q},\mathbf{p}$ is also optimal. We repeat the process on the new optimal distribution till we find one which has only 1 or 2 non-zero entries. ■

We next show how to choose the labels $l_1,\ldots,l_m$ using the distributions $\mathbf{p}_i$. For each i, let $\left\{l_i^+, l_i^-\right\}$ be the support of $\mathbf{p}_i$ so that

$$C\left(i,l_i^+\right) \le \mathbb{E}_{l\sim\mathbf{p}_i}[C(i,l)] \le C\left(i,l_i^-\right).$$

(When $\mathbf{p}_i$ has only one non-zero element, then $l_i^+ = l_i^-$.) For brevity, we use p_i^+ and p_i^- to denote $p_i\left(l_i^+\right)$ and $p_i\left(l_i^-\right)$, respectively. If the costs of both labels are equal, we assume without loss of generality that $\mathbf{p}_i$ is concentrated on label l_i^-:

$$C\left(i,l_i^-\right) - C\left(i,l_i^-\right) = 0 \implies p_i^+ = 0, p_i^- = 1. \tag{78}$$

We will choose each label l_i from the set $\left\{l_i^-, l_i^+\right\}$. In fact, we will choose a partition S_+, S_- of the examples $1, \ldots, m$ and choose the label depending on which side S_ξ, for $\xi \in \{-,+\}$, of the partition element i belongs to:

$$l_i = l_i^\xi \text{ if } i \in S_\xi.$$

In order to guide our choice for the partition, we introduce parameters a_i, b_i as follows:

$$\begin{aligned} a_i &= C(i, l_i^-) - C(i, l_i^+), \\ b_i &= \phi_{T-t-1}^{\mathbf{B}(i)}\left(\mathbf{s}_t(i) + \mathbf{e}_{l_i^-}\right) - \phi_{T-t-1}^{\mathbf{B}(i)}\left(\mathbf{s}_t(i) + \mathbf{e}_{l_i^+}\right). \end{aligned}$$

Notice that for each example i and each sign-bit $\xi \in \{-1,+1\}$, we have the following relations:

$$C(i, l_i^\xi) = \mathbb{E}_{l \sim \mathbf{p}_i}[C(i,l)] - \xi(1 - p_i^\xi) a_i \tag{79}$$

$$\phi_{T-t-1}^{\mathbf{B}(i)}\left(\mathbf{s}_t(i) + \mathbf{e}_{l_i^\xi}\right) = \mathbb{E}_{l \sim \mathbf{p}_i}\left[\phi_{T-t}^{\mathbf{B}(i)}(i,l)\right] - \xi(1 - p_i^\xi) b_i. \tag{80}$$

Then the cost incurred by the choice of labels can be expressed in terms of the parameters a_i, b_i as follows:

$$\begin{aligned} \sum_{i \in S_+} C(i, l_i^+) + \sum_{i \in S_-} C(i, l_i^-) &= \sum_{i \in S_+}\left\{\mathbb{E}_{l \sim \mathbf{p}_i}[C(i,l)] - a_i + p_i^+ a_i\right\} \\ &\quad + \sum_{i \in S_-}\left\{\mathbb{E}_{l \sim \mathbf{p}_i}[C(i,l)] + p_i^+ a_i\right\} \\ &= \sum_{i=1}^m \mathbb{E}_{l \sim \mathbf{p}_i}[C(i,l)] + \left(\sum_{i=1}^m p_i^+ a_i - \sum_{i \in S_+} a_i\right) \\ &\le \sum_{i=1}^m \langle \mathbf{C}(i), \mathbf{B}(i) \rangle + \left(\sum_{i=1}^m p_i^+ a_i - \sum_{i \in S_+} a_i\right), \end{aligned} \tag{81}$$

where the first equality follows from (79), and the inequality follows from the constraint on $\mathbf{p}_i$ in (77). Similarly, the potential of the new states is given by

$$\begin{aligned} &\sum_{i \in S_+} \phi_{T-t-1}^{\mathbf{B}(i)}\left(\mathbf{s}_t(i) + \mathbf{e}_{l_i^+}\right) + \sum_{i \in S_-} \phi_{T-t-1}^{\mathbf{B}(i)}\left(\mathbf{s}_t(i) + \mathbf{e}_{l_i^-}\right) \\ &= \sum_{i \in S_+}\left\{\mathbb{E}_{l \sim \mathbf{p}_i}\left[\phi_{T-t-1}^{\mathbf{B}(i)}(\mathbf{s}_t(i) + \mathbf{e}_l)\right] - b_i + p_i^+ b_i\right\} \\ &\quad + \sum_{i \in S_-}\left\{\mathbb{E}_{l \sim \mathbf{p}_i}\left[\phi_{T-t-1}^{\mathbf{B}(i)}(\mathbf{s}_t(i) + \mathbf{e}_l)\right] + p_i^+ b_i\right\} \\ &= \sum_{i=1}^m \mathbb{E}_{l \sim \mathbf{p}_i}\left[\phi_{T-t-1}^{\mathbf{B}(i)}(\mathbf{s}_t(i) + \mathbf{e}_l)\right] + \left(\sum_{i=1}^m p_i^+ b_i - \sum_{i \in S_+} b_i\right) \\ &= \sum_{i=1}^m \phi_{T-t}^{\mathbf{B}(i)}(\mathbf{s}_t(i)) + \left(\sum_{i=1}^m p_i^+ b_i - \sum_{i \in S_+} b_i\right), \end{aligned} \tag{82}$$

where the first equality follows from (80), and the last equality from an optimal choice of $\mathbf{p}_i$ satisfying (75). Now, (81) and (82) imply that in order to satisfy (73) and (74), it suffices to choose a subset S_+ satisfying

$$\sum_{i\in S_+} a_i \geq \sum_{i=1}^{m} p_i^+ a_i, \qquad\qquad \sum_{i\in S_+} b_i \leq \frac{m\varepsilon}{T} + \sum_{i=1}^{m} p_i^+ b_i. \tag{83}$$

We simplify the required conditions. Notice the first constraint tries to ensure that S_+ is big, while the second constraint forces it to be small, provided the b_i are non-negative. However, if $b_i < 0$ for any example i, then adding this example to S_+ only helps both inequalities. In other words, if we can always construct a set S_+ satisfying (83) in the case where the b_i are non-negative, then we may handle the more general situation by just adding the examples i with negative b_i to the set S_+ that would be constructed by considering only the examples $\{i : b_i \geq 0\}$. Therefore we may assume without loss of generality that the b_i are non-negative. Further, assume (by relabeling if necessary) that $a_1,\ldots,a_{m'}$ are positive and $a_{m'+1},\ldots a_m = 0$, for some $m' \leq m$. By (78), we have $p_i^+ = 0$ for $i > m'$. Therefore, by assigning the examples $m'+1,\ldots,m$ to the opposite partition S_-, we can ensure that (83) holds if the following is true:

$$\sum_{i\in S_+} a_i \;\geq\; \sum_{i=1}^{m'} p_i^+ a_i, \tag{84}$$

$$\sum_{i\in S_+} b_i \;\leq\; \max_{i=1}^{m'} |b_i| + \sum_{i=1}^{m'} p_i^+ b_i, \tag{85}$$

where, for (85), we additionally used that, by the choice of m (25) and the bound on loss variation (24), we have

$$\frac{m\varepsilon}{T} \geq \varnothing(L,T) \geq b_i \text{ for } i = 1,\ldots,m.$$

The next lemma shows how to construct such a subset S_+, and concludes our lower bound proof.

Lemma 32 *Suppose $a_1,\ldots,a_{m'}$ are positive and $b_1,\ldots,b_{m'}$ are non-negative reals, and $p_1^+,\ldots,p_{m'}^+ \in [0,1]$ are probabilities. Then there exists a subset $S_+ \subseteq \{1,\ldots,m'\}$ such that* (84) *and* (85) *hold.*

Proof Assume, by relabeling if necessary, that the following ordering holds:

$$\frac{a(1)-b(1)}{a(1)} \geq \cdots \geq \frac{a(m')-b(m')}{a(m')}. \tag{86}$$

Let $I \leq m'$ be the largest integer such that

$$a_1 + a_2 + \cdots + a_I < \sum_{i=1}^{m'} p_i^+ a_i. \tag{87}$$

Since the p_i^+ are at most 1, I is in fact at most $m'-1$. We will choose S_+ to be the first $I+1$ examples $S_+ = \{1,\ldots,I+1\}$. Observe that (84) follows immediately from the definition of I. Further, (85) will hold if the following is true

$$b_1 + b_2 + \cdots + b_I \leq \sum_{i=1}^{m'} p_i^+ b_i, \tag{88}$$

since the addition of one more example $I+1$ can exceed this bound by at most $b_{I+1} \leq \max_{i=1}^{m'} |b_i|$. We prove (88) by showing that the left hand side of this equation is not much more than the left hand side of (87). We first rewrite the latter summation differently. The inequality in (87) implies we can pick $\tilde{p}_1^+, \ldots, \tilde{p}_{m'}^+ \in [0,1]$ (e.g., by simply scaling the p_i^+'s appropriately) such that

$$\begin{aligned} a_1 + \ldots + a_I &= \sum_{i=1}^{m'} \tilde{p}_i^+ a_i && (89) \\ \text{for } i = 1, \ldots, m'\text{: } \tilde{p}_i^+ &\leq p_i. && (90) \end{aligned}$$

By subtracting off the first I terms in the right hand side of (89) from both sides we get

$$(1 - \tilde{p}_1^+)a_1 + \cdots + (1 - \tilde{p}_I^+)a_I = \tilde{p}_{I+1}^+ a_{I+1} + \cdots + \tilde{p}_{m'}^+ a_{m'}.$$

Since the terms in the summations are non-negative, we may combine the above with the ordering property in (86) to get

$$\begin{aligned} & (1 - \tilde{p}_1^+)a_1 \left(\frac{a_1 - b_1}{a_1} \right) + \cdots + (1 - \tilde{p}_I^+)a_I \left(\frac{a_I - b_I}{a_I} \right) \\ \geq \; & \tilde{p}_{I+1}^+ a_{I+1} \left(\frac{a_{I+1} - b_{I+1}}{a_{I+1}} \right) + \cdots + \tilde{p}_{m'}^+ a_{m'} \left(\frac{a_{m'} - b_{m'}}{a_{m'}} \right). \qquad (91) \end{aligned}$$

Adding the expression

$$\tilde{p}_1^+ a_1 \left(\frac{a_1 - b_1}{a_1} \right) + \cdots + \tilde{p}_I^+ a_I \left(\frac{a_I - b_I}{a_I} \right)$$

to both sides of (91) yields

$$\begin{aligned} \sum_{i=1}^{I} a_i \left(\frac{a_i - b_i}{a_i} \right) &\geq \sum_{i=1}^{m'} \tilde{p}_i^+ a_i \left(\frac{a_i - b_i}{a_i} \right) \\ \text{i.e., } \sum_{i=1}^{I} a_i - \sum_{i=1}^{I} b_i &\geq \sum_{i=1}^{m'} \tilde{p}_i^+ a_i - \sum_{i=1}^{m'} \tilde{p}_i^+ b_i \\ \text{i.e., } \sum_{i=1}^{I} b_i &\leq \sum_{i=1}^{m'} \tilde{p}_i^+ b_i, && (92) \end{aligned}$$

where the last inequality follows from (89). Now (88) follows from (92) using (90) and the fact that the b_i's are non-negative. ■

This completes the proof of the lower bound.

A.2 Consistency Proofs

Here we sketch the proofs of Lemmas 26 and 27. Our approach will be to relate our algorithm to AdaBoost and then use relevant known results on the consistency of AdaBoost. We first describe the correspondence between the two algorithms, and then state and connect the relevant results on AdaBoost to the ones in this section.

For any given multiclass data set and weak classifier space, we will obtain a transformed binary data set and weak classifier space, such that the run of AdaBoost.MM on the original data set will be in perfect correspondence with the run of AdaBoost on the transformed data set. In particular, the loss and error on both the training and test set of the combined classifiers produced by our algorithm will be exactly equal to those produced by AdaBoost, while the space of functions and classifiers on the two data sets will be in correspondence.

Intuitively, we transform our multiclass classification problem into a single binary classification problem in a way similar to the all-pairs multiclass to binary reduction. A very similar reduction was carried out by Freund and Schapire (1997). Borrowing their terminology, the transformed data set roughly consists of *mislabel* triples (x,y,l) where y is the true label of the example and l is an incorrect example. The new binary label of a mislabel triple is always -1, signifying that l is not the true label. A multiclass classifier becomes a binary classifier that predict ± 1 on the mislabel triple (x,y,l) depending on whether the prediction on x matches label l; therefore error on the transformed binary data set is low whenever the multiclass accuracy is high. The details of the transformation are provided in Figure 12.

Some of the properties between the functions and their transformed counterparts are described in the next lemma, showing that we are essentially dealing with similar objects.

Lemma 33 *The following are identities for any scoring function $F : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ and weight function $\alpha : \mathcal{H} \to \mathbb{R}$:*

$$\begin{aligned} \widehat{\text{risk}}\left(F_\alpha\right) &= \widetilde{\widehat{\text{risk}}}\left(\widetilde{F}_{\widetilde{\alpha}}\right) & (93) \\ \text{risk}_D\left(\bar{F}\right) &= \widetilde{\text{risk}_D}\left(\bar{\widetilde{F}}\right). & (94) \end{aligned}$$

The proofs involve doing straightforward algebraic manipulations to verify the identities and are omitted.

The next lemma connects the two algorithms. We show that the scoring function output by AdaBoost when run on the transformed data set is the transformation of the function output by our algorithm. The proof again involves tedious but straightforward checking of details and is omitted.

Lemma 34 *If AdaBoost.MM produces scoring function F_α when run for T rounds with the training set S and weak classifier space $\mathcal{H}$, then AdaBoost produces the scoring function $\widetilde{F}_{\widetilde{\alpha}}$ when run for T rounds with the training set $\widetilde{S}$ and space $\widetilde{\mathcal{H}}$. We assume that for both the algorithms, Weak Learner returns the weak classifier in each round that achieves the maximum edge. Further we consider the version of AdaBoost.MM that chooses weights according to the approximate rule* (61)*.*

We next state the result for AdaBoost corresponding to Lemma 26 , which appears in Mukherjee et al. (2011).

Lemma 35 (Theorem 8 in Mukherjee et al. (2011)) *Suppose AdaBoost produces the scoring function $\widetilde{F}_{\widetilde{\alpha}}$ when run for T rounds with the training set $\widetilde{S}$ and space $\widetilde{\mathcal{H}}$. Then*

$$\widetilde{\widehat{\text{risk}}}\left(\widetilde{F}_{\widetilde{\alpha}}\right) \leq \inf_{\widetilde{\beta}:\widetilde{\mathcal{H}}\to\mathbb{R}} \widetilde{\widehat{\text{risk}}}\left(\widetilde{F}_{\widetilde{\beta}}\right) + C/T,$$

where the constant C depends only on the data set.

The previous lemma, along with (93) immediately proves Lemma 26. The result for AdaBoost corresponding to Lemma 27 appears in Schapire and Freund (2012).

	AdaBoost.MM	AdaBoost
Labels	$\mathcal{Y} = \{1, \ldots, k\}$	$\widetilde{\mathcal{Y}} = \{-1, +1\}$
Examples	$\mathcal{X}$	$\widetilde{\mathcal{X}} = \mathcal{X} \times ((\mathcal{Y} \times \mathcal{Y}) \setminus \{(y,y) : y \in \mathcal{Y}\})$
Weak classifiers	$h : \mathcal{X} \to \mathcal{Y}$	$\widetilde{h} : \widetilde{X} \to \{-1, 0, +1\}$, where $\widetilde{h}(x,y,l) = \mathbf{1}[h(x) = l] - \mathbf{1}[h(x) = y]$
Classifier space	$\mathcal{H}$	$\widetilde{\mathcal{H}} = \left\{\widetilde{h} : h \in \mathcal{H}\right\}$
Scoring function	$F : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$	$\widetilde{F} : \widetilde{\mathcal{X}} \to \mathbb{R}$ where $\widetilde{F}(x,y,l) = F(x,l) - F(x,y)$
Clamped function	$\bar{F}(x,y) =$ $\max\{-C, F(x,l) - \max_{l'} F_T(x,l')\}$	$\bar{\widetilde{F}}(x,y,l) = \widetilde{F}(x,y,l)$, if $\lvert\widetilde{F}(x,y,l)\rvert \le C$ $\bar{\widetilde{F}}(x,y,l) = C$, if $\lvert\widetilde{F}(x,y,l)\rvert > C$
Classifier weights	$\alpha : \mathcal{H} \to \mathbb{R}$	$\widetilde{\alpha} : \widetilde{\mathcal{H}} \to \mathbb{R}$ where $\widetilde{\alpha}\left(\widetilde{h}\right) = \alpha(h)$
Combined hypothesis	F_α where $F_\alpha(x,l) = \sum_{h \in \mathcal{H}} \alpha(h) \mathbf{1}[h(x) = l]$	$\widetilde{F}_{\widetilde{\alpha}}$ where $\widetilde{F}_{\widetilde{\alpha}}(x,y,l) = \sum_{\widetilde{h} \in \widetilde{\mathcal{H}}} \widetilde{\alpha}\left(\widetilde{h}\right) \widetilde{h}(x,y,l)$
Training set	$S = \{(x_i, y_i) : 1 \le i \le m\}$	$\widetilde{S} =$ $\{((x_i, y_i, l), \xi) : \xi = -1, l \neq y_i, 1 \le i \le m\}$
Test distribution	D over $\mathcal{X} \times \mathcal{Y}$	$\widetilde{D}$ over $\widetilde{X} \times \widetilde{Y}$ where $\widetilde{D}((x,y,l),-1) = D(x,y)/(k-1)$ $\widetilde{D}((x,y,l),+1) = 0$
Empirical risk	$\widehat{\text{risk}}(F) =$ $\frac{1}{m} \sum_{i=1}^{m} \sum_{l \neq y_i} e^{F(x_i,l) - F(x_i,y_i)}$	$\widetilde{\widehat{\text{risk}}}\left(\widetilde{F}\right)$ $\frac{1}{m(k-1)} \sum_{i=1}^{m} \sum_{l \neq y_i} e^{-\xi \widetilde{F}(x_i,y_i,l)}$
Test risk	$\text{risk}_D(F) =$ $\mathbb{E}_{(x,y) \sim D}\left[\sum_{l \neq y} e^{F(x,l) - F(x,y)}\right]$	$\widetilde{\text{risk}_D}\left(\widetilde{F}\right) =$ $\mathbb{E}_{((x,y,l),\xi) \sim \widetilde{D}}\left[e^{-\xi \widetilde{F}(x,y,l)}\right]$

Figure 12: Details of transformation between AdaBoost.MM and AdaBoost.

Lemma 36 (Theorem 12.2 in Schapire and Freund (2012)) *Suppose AdaBoost produces the scoring function $\widetilde{F}$ when run for $T = \sqrt{m}$ rounds with the training set $\widetilde{S}$ and space $\widetilde{\mathcal{H}}$. Then*

$$\Pr\left[\text{risk}_D\left(\bar{\widetilde{F}}\right) \le \inf_{\widetilde{F}' : \widetilde{\mathcal{X}} \to \mathbb{R}} \text{risk}_D(\widetilde{F}') + O\left(m^{-c}\right)\right] \ge 1 - \frac{1}{m^2},$$

where the constant C depends only on the data set.

The proof of Lemma 27 follows immediately from the above lemma and (94).

References

Jacob Abernethy, Peter L. Bartlett, Alexander Rakhlin, and Ambuj Tewari. Optimal stragies and minimax lower bounds for online convex games. In *COLT*, pages 415–424, 2008.

Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

Peter L. Bartlett and Mikhail Traskin. AdaBoost is consistent. *Journal of Machine Learning Research*, 8:2347–2368, 2007.

Peter L. Bartlett, Michael I. Jordan, and Jon D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, March 2006.

Alina Beygelzimer, John Langford, and Pradeep Ravikumar. Error-correcting tournaments. In *Algorithmic Learning Theory: 20th International Conference*, pages 247–262, 2009.

Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, January 1995.

Günther Eibl and Karl-Peter Pfeiffer. Multiclass boosting for weak classifiers. *Journal of Machine Learning Research*, 6:189–210, 2005.

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

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

Yoav Freund and Manfred Opper. Continuous drifting games. *Journal of Computer and System Sciences*, pages 113–132, 2002.

Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156, 1996a.

Yoav Freund and Robert E. Schapire. Game theory, on-line prediction and boosting. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, pages 325–332, 1996b.

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

Trevor Hastie and Robert Tibshirani. Classification by pairwise coupling. *Annals of Statistics*, 26 (2):451–471, 1998.

Vladimir Koltchinskii and Dmitriy Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *Annals of Statistics*, 30(1), February 2002.

Ping Li. Robust logitboost and adaptive base class (abc) logitboost. In *UAI*, pages 302–311, 2010.

Philip M. Long and Rocco A. Servedio. Random classification noise defeats all convex potential boosters. *Machine Learning*, 78:287–304, 2010.

Indraneel Mukherjee and Robert E. Schapire. Learning with continuous experts using drifting games. *Theoretical Computer Science*, 411(29-30):2670–2683, June 2010.

Indraneel Mukherjee, Cynthia Rudin, and Robert E. Schapire. The rate of convergence of AdaBoost. In *The 24th Annual Conference on Learning Theory*, 2011.

Gunnar Rätsch and Manfred K. Warmuth. Efficient margin maximizing with boosting. *Journal of Machine Learning Research*, 6:2131–2152, 2005.

R. Tyrrell Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

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

Robert E. Schapire. Drifting games. *Machine Learning*, 43(3):265–291, June 2001.

Robert E. Schapire and Yoav Freund. *Boosting: Foundations and Algorithms*. MIT Press, 2012.

Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, December 1999.

Robert E. Schapire and Yoram Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, May/June 2000.

Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26(5):1651–1686, October 1998.

Ambuj Tewari and Peter L. Bartlett. On the Consistency of Multiclass Classification Methods. *Journal of Machine Learning Research*, 8:1007–1025, May 2007.

Tong Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, 32(1):56–134, 2004.

Ji Zhu, Hui Zou, Saharon Rosset, and Trevor Hastie. Multi-class AdaBoost. *Statistics and Its Interface*, 2:349360, 2009.