A Theory of Multiclass Boosting

Indraneel Mukherjee **Robert E. Schapire**

Princeton University, Department of Computer Science, Princeton, NJ 08540
{imukherj,schapire}@cs.princeton.edu

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.

1 Introduction

Boosting [17] 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 [6] algorithm is optimal in a certain sense, and that AdaBoost [11] 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, e.g. 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 [9]. The most common approaches so far have relied on reductions to binary classification [2], 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 [19], 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 [9], 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 Zhu et al.'s SAMME algorithm [21] 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 [19, 9] (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 [8, 16], we are able to compute the optimal strategy for the games arising out of each weak-learning condition in the family described above. These optimal strategies have a natural interpretation in terms of random walks, a phenomenon that has been observed in other settings [1, 6].

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. 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 [18, 11, 13]. (We omit these due to lack of space.)

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 one of the game-theoretic strategies.

We present experiments aimed at testing the efficacy of the new methods 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.

Previous work. The first boosting algorithms were given by Schapire [15] and Freund [6], followed by their AdaBoost algorithm [11]. Multiclass boosting techniques include AdaBoost.M1 and AdaBoost.M2 [11], as well as AdaBoost.MH and AdaBoost.MR [19]. Other approaches include [5, 21]. There are also more general approaches that can be applied to boosting including [2, 3, 4, 12]. Two game-theoretic perspectives have been applied to boosting. The first one [10, 14] views the weak-

learning condition as a minimax game, while drifting games [16, 6] were designed to analyze the most efficient boosting algorithms. These games have been further analyzed in the multiclass and continuous time setting in [8].

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. Inner product of two vectors $\mathbf{u}, \mathbf{v}$ is denoted by $\langle \mathbf{u}, \mathbf{v} \rangle$. The Frobenius inner product of two matrices $\mathrm{Tr}(\mathbf{M}\mathbf{M}')$ will be denoted by $\mathbf{M} \bullet \mathbf{M}'$. The indicator function is denoted by $\mathbb{1}[\cdot]$. The distribution over the set $\{1, \ldots, k\}$ will be denoted by $\Delta\{1, \ldots, k\}$.

In multiclass classification, we want to predict the labels of examples lying in some set X. Each example $x \in X$ has a unique y label in the set $\{1, \ldots, k\}$, where $k \geq 2$. We are provided a training set of labeled examples $\{(x_1, y_1), \ldots, (x_m, y_m)\}$.

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 classifier 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.

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: (1) 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 $C(i,l)$. The cost-matrix may not be arbitrary, but should conform to certain restrictions as discussed below. (2) 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 $\mathbb{1}[h(i) = j]$. (3) 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:

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

By carefully choosing the cost matrices in each round, Booster aims to minimize the training error of the final classifer 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 classfier 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: $\forall i : C(i, y_i) \leq C(i, \bar{y}_i)$, while the restriction on the returned weak classifier h requires less cost than predicting randomly: $\sum_i C(i, h(x_i)) \leq \sum_i \left\{ \left(\frac{1}{2} - \frac{\gamma}{2}\right) C(i, \bar{y}_i) + \left(\frac{1}{2} + \frac{\gamma}{2}\right) C(i, y_i) \right\}$. 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 $C(i,1) \leq C(i,2)$. 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: $\forall \mathbf{C} \in \mathcal{C}^{\text{bin}}, \exists h \in \mathcal{H} : \mathbf{C} \bullet \left(\mathbf{1}_h - \mathbf{U}^{\text{bin}}_{\gamma}\right) \leq \mathbf{0}$. 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 $k \geq 2$ as we now show.

Let $\mathcal{C} \subseteq \mathbb{R}^{m\times k}$ and matrix $\mathbf{B} \in \mathbb{R}^{m\times k}$, 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 [9], AdaBoost.MH [19] and AdaBoost.MR [9, 19] (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 eligible for boosting, or simply *boostable*, if there exists a distribution $\boldsymbol{\lambda}$ on this space that linearly separates the data: $\forall i : \operatorname{argmax}_{l\in\{1,\ldots,k\}} \sum_{h\in\mathcal{H}} \lambda(h) \mathbb{1}\left[h(x_i) = l\right] = y_i$. 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*, i.e. certain boostable spaces will fail to satisfy the conditions. Or it could be *too weak* i.e., 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. In the next section we will describe conditions captured by our framework that avoid being too weak or too strong.

3 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 $1/2 + \gamma$. Further, this is the weakest condition under which boosting is possible as follows from a game-theoretic perspective [10, 14] . Multiclass weak-learning conditions with similar properties are missing in the literature. In this section we show how our framework captures such 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.

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, i.e., 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.

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

Theorem 1 (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.*

The proof involves the Von-Neumann Minimax theorem, and is in the spirit of the ones in [10]. 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 2 (Relaxed necessity)**.** *For every boostable weak classifier space $\mathcal{H}$, there exists a $\gamma > 0$ and $\mathbf{B} \in \mathcal{B}^{eor}_{\gamma}$ such that $\mathcal{H}$ satisfies the weak-learning condition $(\mathcal{C}^{eor}, \mathbf{B})$.*

The proof shows existence through non-constructive averaging arguments. Theorem 2 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 5 suggest $\left(\mathcal{C}^{\text{eor}}, \mathbf{U}_{\gamma}\right)$ is effective with very simple weak-learners compared to popular boosting algorithms. (Here $\mathbf{U}_{\gamma} \in \mathcal{B}^{\text{eor}}_{\gamma}$ 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 (supplement).

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 $\mathbf{B} \in \mathcal{B}^{\text{eor}}_{\gamma}$, but with the *worst* of them:

$$\forall \mathbf{C} \in \mathcal{C}^{\text{eor}}, \exists h \in \mathcal{H} : \mathbf{C} \bullet \mathbf{1}_h \leq \max_{\mathbf{B} \in \mathcal{B}^{\text{eor}}_{\gamma}} \mathbf{C} \bullet \mathbf{B}. \tag{3}$$

Condition (3) states that during the course of the same boosting game, Weak-Learner may choose to beat *any* edge-over-random baseline $\mathbf{B} \in \mathcal{B}^{\text{eor}}_{\gamma}$, 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 [19, 9]. This is perhaps remarkable since the latter is based on the apparently completely unrelated all-pairs multiclass to binary reduction: the MR condition is given by $(\mathcal{C}^{\text{MR}}, \mathbf{B}^{\text{MR}}_{\gamma})$, 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 $\mathbf{B}^{\text{MR}}_{\gamma} \in \mathbb{R}^{m \times k}$ is the matrix that has γ on the first column and $-\gamma$ on all other columns(supplement). Further, the MR condition, and hence (3), can be shown to be neither too weak nor too strong.

Theorem 3 (MR)**.** *A weak classifier space $\mathcal{H}$ satisfies AdaBoost.MR's weak-learning condition $(\mathcal{C}^{MR}, \mathbf{B}^{MR}_{\gamma})$ if and only if it satisfies* (3)*. Moreover, this condition is equivalent to being boostable.*

Next, we illustrate the strengths of our random-over-edge weak-learning conditions through concrete comparisons with previous algorithms.

Comparison with SAMME. The SAMME algorithm of [21] requires the weak classifiers to achieve less error than uniform random guessing for multiple labels; in our language, their weak-learning condition is $(\mathcal{C} = \{(-t, t, t, \ldots) : t \geq 0\}, \mathbf{U}_{\gamma})$. As is well-known, this condition is not sufficient for boosting to be possible. In particular, consider the dataset $\{(a,1),(b,2)\}$ with $k = 3, m = 2$, and a weak classifier space consisting of h_1, h_2 which always predict $1, 2$, respectively. 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 is $\mathbf{C} = (\mathbf{c}(1) = (-1, +1, 0), \mathbf{c}(2) = (+1, -1, 0)) \in \mathcal{C}^{\text{eor}}$, both classifiers in the above example suffer more loss than the random player $\mathbf{U}_{\gamma}$, and fail to satisfy our condition.

Comparison with AdaBoost.MH. AdaBoost.MH is a popular multiclass boosting algorithm that is based on the one-against-all reduction[19]. However, we show that its implicit demands on the weak classifier space is too strong. 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.

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 can be shown (supplement) that AdaBoost.MH's weak-learning condition is the pair $(\mathcal{C}^{\text{MH}}, \mathbf{B}^{\text{MH}}_{\gamma})$, where $\mathcal{C}^{\text{MH}}$ has non-(positive)negative entries on (in)correct labels, and where each row of the matrix $\mathbf{B}^{\text{MH}}_{\gamma}$ 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 $\mathbf{C} \in \mathcal{C}^{\text{MH}}$ with -1 in the first column and zeroes elsewhere, $\mathbf{C} \bullet \left(\mathbf{1}_h - \mathbf{B}_\gamma^{\text{MH}}\right) = 1/2 - 1/k$. This is positive when $k > 2$, so that $\mathcal{H}$ fails to satisfy AdaBoost.MH's condition.

4 Algorithms

In this section we devise algorithms by analyzing the boosting games that employ our edge-over-random weak-learning conditions. We compute the optimum Booster strategy against a completely adversarial Weak-Learner, which here is permitted to choose weak classifiers without restriction, i.e. the entire space $\mathcal{H}^{\text{all}}$ 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 [16] for solving boosting games, in turn inspired by Freund's Boost-by-majority algorithm [6], which we review next.

The OS Algorithm. Fix the number of rounds T and an edge-over-random weak-learning condition $(\mathcal{C}, \mathbf{B})$. For simplicity of presentation we fix the weights $\alpha_t = 1$ in each round. With $\mathbf{f}_T$ defined as in (1), the optimum Booster payoff can be written as

$$\min_{\mathbf{C}_1 \in \mathcal{C}} \max_{\substack{h_1 \in \mathcal{H}^{\text{all}}: \\ \mathbf{C}_1 \bullet \left(\mathbf{1}_{h_1} - \mathbf{B}\right) \leq \mathbf{0}}} \cdots \min_{\mathbf{C}_T \in \mathcal{C}} \max_{\substack{h_T \in \mathcal{H}^{\text{all}}: \\ \mathbf{C}_T \bullet \left(\mathbf{1}_{h_T} - \mathbf{B}\right) \leq \mathbf{0}}} (1/m) \sum_{i=1}^{m} L(f_T(x_i, 1), f_T(x_i, 2), \ldots, f_T(x_i, k)).$$

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

Directly analyzing the optimal payoff is hard. However, Schapire [16] 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:

$$\phi_0^{\mathbf{b}} = L; \quad \phi_t^{\mathbf{b}}(\mathbf{s}) = \min_{\mathbf{c} \in \mathbb{R}^k : \forall l : c(1) \leq c(l)} \max_{\mathbf{p} \in \Delta\{1,\ldots,k\}} \left\{ \mathbb{E}_{l \sim \mathbf{p}} \left[\phi_{t-1}^{\mathbf{b}} \left(\mathbf{s} + \mathbf{e}_l \right) \right] : \mathbb{E}_{l \sim \mathbf{p}} \left[c(l) \right] \leq \langle \mathbf{b}, \mathbf{c} \rangle \right\}, \quad (4)$$

where $\mathbf{e}_l \in \mathbb{R}^k$ is the unit-vector whose lth coordinate is 1 and the remaining coordinates zero. These potential functions compute an estimate $\phi_t^{\mathbf{b}}(\mathbf{s}_t)$ 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 $s_t(l) = \sum_{t'=1}^{t-1} \mathbb{1}\left[h_{t'}(x) = l\right]$, and the number of rounds t remaining. Using these functions, Schapire [16] proposed a Booster strategy, aka the OS strategy, which, in round t, constructs a cost matrix $\mathbf{C} \in \mathcal{C}$, whose each row $\mathbf{C}(i)$ achieves the minimum of the right hand side of (4) with $\mathbf{b}$ replaced by $\mathbf{B}(i)$, t replaced by $T - t$, and $\mathbf{s}$ replaced by current state $\mathbf{s}_t(i)$. The following theorem 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.

Theorem 4 (Extension of results in [16])**.** *Suppose the weak-learning condition is given by $(\mathcal{C}, \mathbf{B})$, If Booster employs the OS algorithm, then the average potential of the states $(1/m) \sum_{i=1}^{m} \phi_t^{\mathbf{B}(i)}(\mathbf{s}(i))$ never increases in any round. In particular, loss suffered after T rounds of play is at most $(1/m) \sum_{i=1}^{m} \phi_T^{\mathbf{B}(i)}(\mathbf{0})$. Further, for any $\epsilon > 0$, when the loss function satisfies some mild conditions, and $m \gg T, k, 1/\epsilon$, no Booster strategy can achieve loss ϵ less than the above bound in T rounds.*

Computing the potentials. In order to implement the OS strategy using our weak-learning conditions, we only need to compute the potential $\phi_t^{\mathbf{b}}$ for distributions $\mathbf{b} \in \Delta\{1, \ldots, k\}$. Fortunately, these potentials have a very simple solution in terms of the *homogeneous* random-walk $\mathcal{R}_{\mathbf{b}}^t(\mathbf{x})$, the random 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 $b(l)$.

Theorem 5. *If L is proper, and $\mathbf{b} \in \Delta\{1, \ldots, k\}$ satisfies $\forall l : b(1) \geq b(l)$, then $\phi_t^{\mathbf{b}}(\mathbf{s}) = \mathbb{E}\left[L\left(\mathcal{R}_{\mathbf{b}}^t(\mathbf{s})\right)\right]$. Furthermore, the vector achieving the minimum in the right hand side of* (4) *is given by $c(l) = \phi_{t-1}^{\mathbf{b}}(\mathbf{s} + \mathbf{e}_l)$.*

Theorem (5) implies the OS strategy chooses the following cost matrix in round t: $c(i, l) = \phi_{T-t-1}^{\mathbf{b}(i)}(\mathbf{s}_t(i) + \mathbf{e}_l)$, where $\mathbf{s}_t(i)$ is the state of example i in round t. Therefore everything boils

down to computing the potentials, which is made possible by Theorem 5. There is no simple closed form solution for the non-convex 0-1 loss $L(\mathbf{s}) = \mathbb{1}[s_1 \leq (\max_{i>1} s_i)]$. However, using Theorem 4, we can write the potential $\phi_t(\mathbf{s})$ explicitly, and then compute it using dynamic programming in $O(t^3 k)$ time. This yields very tight bounds.

To obtain a more efficient procedure, and one that we will soon show can be made adaptive, we next focus on the exponential loss associated with AdaBoost that does have a closed form solution.

Lemma 1. *If $L(\mathbf{s}) = \exp(\eta_2(s_2 - s_1)) + \cdots + \exp(\eta_k(s_k - s_1))$, where each η_l is positive, then the solution in Theorem 5 evaluates to $\phi_t^{\mathfrak{b}}(\mathbf{s}) = \sum_{l=2}^{k} (a_l)^t e^{\eta_l (s_l - s_1)}$, where $a_l = 1 - (b_1 + b_l) + e^{\eta_l} b_l + e^{-\eta_l} b_1$.*

The proof by induction is straightforward. In particular, when the condition is $(\mathcal{C}^{\text{eor}}, \mathbf{U}_\gamma)$ and $\boldsymbol{\eta} = (\eta, \eta, \ldots)$, the relevant potential is $\phi_t(\mathbf{s}) = \kappa(\gamma, \eta)^t \sum_{l=2}^{k} e^{\eta(s_l - s_1)}$ where $\kappa(\gamma, \eta) = 1 + \frac{(1-\gamma)}{k}(e^{\eta} + e^{-\eta} - 2) - (1 - e^{-\eta})\gamma$. 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

$$c(i,l) = \begin{cases} (e^{\eta} - 1)\, e^{\eta(s_l - s_1)} & \text{if } l > 1, \\ (e^{-\eta} - 1) \sum_{j=2}^{k} e^{\eta(s_j - s_1)} & \text{if } l = 1, \end{cases} \tag{5}$$

With such a choice, Theorem 4 and the form of the potential guarantee that the average loss $(1/m)\sum_{i=1}^{m} L(\mathbf{s}_t(i))$ of the states $\mathbf{s}_t(i)$ changes by a factor of at most $\kappa(\gamma, \eta)$ every round. Hence the final loss is at most $(k-1)\kappa(\gamma, \eta)^T$.

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 attempt to fix this via two approaches: prescribing a decaying sequence of edges $\gamma_1, \ldots, \gamma_T$, or being completely flexible, aka *adaptive*, with respect to the edges returned by the weak-learner. In either case, we only use the edge-over-random condition $(\mathcal{C}^{\text{eor}}, \mathbf{U}_\gamma)$, but with varying values of γ.

Fixed sequence of edges. With a prescribed sequence of edges $\gamma_1, \ldots, \gamma_T$ the weak-learning condition $(\mathcal{C}^{\text{eor}}, \mathbf{U}^{\gamma_t})$ in each round t is different. We allow the weights $\alpha_1, \ldots, \alpha_T$ to be arbitrary, but they must be fixed in advance. All the results for uniform γ and weights $\alpha_t = 1$ hold in this case as well. In particular, by the arguments leading to (5), if we want to minimize $\sum_{i=1}^{m}\sum_{l=2}^{k} e^{\{f_t(i,l) - f_t(i,1)\}}$, where f_t is as defined in (1), then the following strategy is optimal: in round t output the cost matrix

$$C(i,l) = \begin{cases} (e^{\alpha_t} - 1)\, e^{f_{t-1}(i,j) - f_{t-1}(i,1)} & \text{if } l > 1, \\ (e^{-\alpha_t} - 1) \sum_{j=2}^{k} e^{f_{t-1}(i,j) - f_{t-1}(i,1)} & \text{if } l = 1. \end{cases} \tag{6}$$

This will ensure that the expression $\sum_{i=1}^{m}\sum_{l=2}^{k} e^{\{f_t(i,l) - f_t(i,1)\}}$ changes by a factor of at most $\kappa(\gamma_t, \alpha_t)$ in each round. Hence the final loss will be at most $(k-1)\prod_{t=1}^{T} \kappa(\gamma_t, \alpha_t)$.

Adaptive. In the adaptive setting, we depart from the game-theoretic framework in that Weak-Learner is no longer adversarial. Further, we are no longer guaranteed to receive a certain sequence of edges. Since the choice of cost-matrix in (6) does not depend on the edges, we could fix an arbitrary set of weights α_t in advance, follow the same algorithm as before and enjoy the same bound $\prod_{t=1}^{T} \kappa(\gamma_t, \alpha_t)$. The trouble with this is $\kappa(\gamma_t, \alpha_t)$ is not less than 1 unless α_t is small compared to γ_t. To ensure progress, the weight α_t must be chosen adaptively as a function of γ_t. Since we do not know what edge we will receive, we choose the cost matrix as before but anticipating infinitesimally small edge, in the spirit of [7], (and with some rescaling)

$$\begin{aligned} C(i,l) &= \lim_{\alpha \to 0} C_\alpha(i,l) \triangleq \frac{1}{\alpha} \begin{cases} (e^{\alpha} - 1)\, e^{f_{t-1}(i,j) - f_{t-1}(i,1)} & \text{if } l > 1, \\ (e^{-\alpha} - 1) \sum_{j=2}^{k} e^{f_{t-1}(i,j) - f_{t-1}(i,1)} & \text{if } l = 1. \end{cases} \\ &= \begin{cases} e^{f_{t-1}(i,j) - f_{t-1}(i,1)} & \text{if } l > 1, \\ -\sum_{j=2}^{k} e^{f_{t-1}(i,j) - f_{t-1}(i,1)} & \text{if } l = 1. \end{cases} \end{aligned} \tag{7}$$

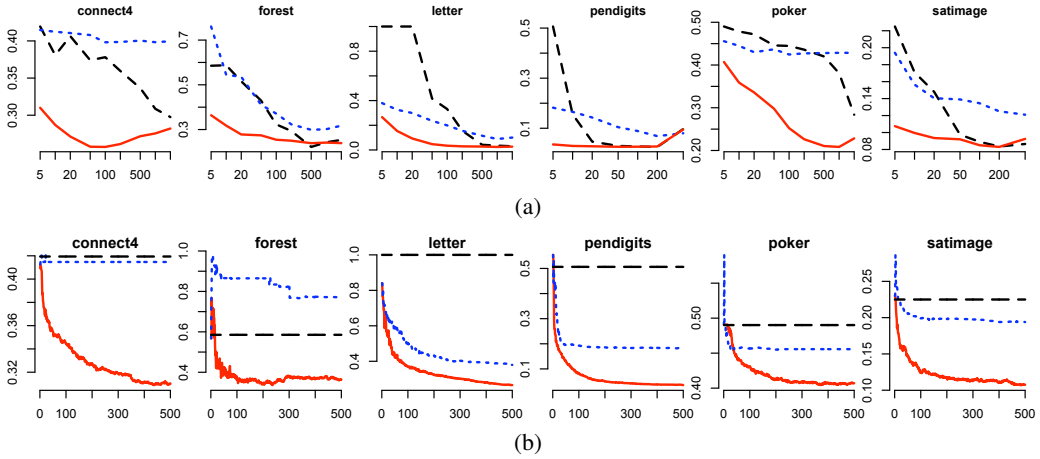

Figure 1: Figure 1(a) plots the final test-errors of `M1`(black, dashed), `MH`(blue, dotted) and `New method`(red, solid) against the maximum tree-sizes allowed as weak classifiers. Figure 1(b) plots how fast the test-errors of these algorithms drop with rounds, when the maximum tree-size allowed is 5.

Since Weak-Learner cooperates, we expect the edge δ_t of the returned classifier h_t on the supplied cost-matrix $\lim_{\alpha\to 0}\mathbf{C}_\alpha$ to be more than just infinitesimal. In that case, by continuity, there are non-infinitesimal choices of the weight α_t such that the edge γ_t achieved by h_t on the cost-matrix $\mathbf{C}_{\alpha_t}$ remains large enough to ensure $\kappa(\gamma_t,\alpha_t)<1$. In fact, with any choice of α_t, we get $\kappa\left(\gamma_t,\alpha_t\right)\leq 1-\frac{1}{2}\left(e^{\alpha_t}-e^{-\alpha_t}\right)\delta_t+\frac{1}{2}\left(e^{\alpha_t}+e^{-\alpha_t}-2\right)$ (supplement). Tuning α_t to $\frac{1}{2}\ln\left(\frac{1+\delta_t}{1-\delta_t}\right)$ results in $\kappa\left(\gamma_t,\alpha_t\right)\leq\sqrt{1-\delta_t^2}$. This algorithm is adaptive, and ensures that the loss, and hence error, after T rounds is at most $(k-1)\prod_{t=1}^{T}\sqrt{1-\delta_t^2}\leq(k-1)\exp\left\{-(1/2)\sum_{t=1}^{T}\delta_t^2\right\}$.

5 Experiments

We report preliminary experimental results on six, varying multiclass UCI datasets.

The first set of experiments were aimed at determining overall performance of our new algorithm. We compared a standard implementation `M1` of AdaBoost.M1 with C4.5 as weak learner, and the Boostexter implementation `MH` of AdaBoost.MH using stumps [20], with the adaptive algorithm described in Section 4, which we call `New method`, using a naive greedy tree-searching algorithm `Greedy` for weak-learner. The size of trees was chosen to be of the same order as the tree sizes used by `M1`. Test errors after 500 rounds of boosting are plotted in Figure 2. 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 compared to C4.5.

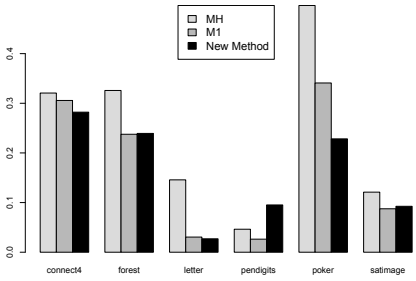

Figure 2: This is a plot of the final test-errors of standard implementations of `M1`, `MH` and `New method` after 500 rounds of boosting.

We next investigated how each algorithm performs with less powerful weak-classifiers, namely, decision trees whose size has been sharply limited to various pre-specified limits. Figure 1(a) shows test-error plotted as a function of tree size. 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. The differences in performance are particularly strong when using the smallest tree sizes.

More insight is provided by plots in Figure 1(b) of the rate of convergence of test 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 cost-matrices, very soon the small-tree learning algorithms are no longer able to meet the excessive requirements of `M1` and `MH`. However, our algorithm makes more reasonable demands that are easily met by the weak learner.

References

[1] Jacob Abernethy, Peter L. Bartlett, Alexander Rakhlin, and Ambuj Tewari. Optimal stragies and minimax lower bounds for online convex games. In *Proceedings of the Nineteenth Annual Conference on Computational Learning Theory*, pages 415–424, 2008.

[2] 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.

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

[4] 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.

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

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

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

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

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

[10] 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, 1996.

[11] 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.

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

[13] V. Koltchinskii and D. Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *Annals of Statistics*, 30(1), February 2002.

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

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

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

[17] Robert E. Schapire. The boosting approach to machine learning: An overview. In *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.

[18] 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.

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

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

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