Proceedings of Machine Learning Research vol 195:1–20, 2023 36th Annual Conference on Learning Theory

Improper Multiclass Boosting

Nataly Brukhim NBRUKHIM@PRINCETON.EDU
Department of Computer Science, Princeton University

Steve Hanneke STEVE.HANNEKE@GMAIL.COM
Department of Computer Science, Purdue University

Shay Moran SMORAN@TECHNION.AC.IL
Departments of Mathematics and Computer Science, Technion and Google Research

Editors: Gergely Neu and Lorenzo Rosasco

Abstract

We study the setting of multiclass boosting with a possibly large number of classes. A recent work by Brukhim, Hazan, Moran, and Schapire, 2021, proved a hardness result for a large class of natural boosting algorithms we call *proper*. These algorithms output predictors that correspond to a plurality-vote aggregation of weak hypotheses. In particular, they showed that proper boosting algorithms must incur a large cost that scales with the number of classes.

In this work we propose an efficient improper multiclass boosting algorithm that circumvents this hardness result. A key component of our algorithm is based on the technique of list learning. In list learning, instead of predicting a single outcome for a given unseen input, the goal is to provide a short menu of predictions. The resulting boosting algorithm has sample and oracle complexity bounds that are entirely independent of the number of classes.

A corollary of the above is that plurality-vote over a learnable class is also learnable. We complement this result by showing that other simple aggregations over hypotheses from a learnable class do *not* preserve learnability, unlike in the binary setting.

Keywords: Multiclass boosting, Compression Schemes, List learning

1. Introduction

Boosting is a fundamental methodology in machine learning that is used to boost the accuracy of weak learning rules into a strong one. Boosting was first studied in the context of binary PAC learning in a line of seminal works which include the celebrated Adaboost algorithm, as well an many other algorithms with various applications (see e.g. Kearns (1988); Schapire (1990); Freund (1990); Freund and Schapire (1997)). The classic weak learning assumption is that when given training examples, the learner outputs a predictor from some $\mathcal{H}$ a class of hypothesis mapping inputs to binary labels, such that its classification error is bounded away below $1/2$. That is, it performs strictly better than random guessing.

The weak learning assumption can be thought of as a *primal* condition for boosting, and is known to be equivalent to a *dual* condition: the data can be approximately represented by a large-margin weighted majority-vote over hypotheses from the class $\mathcal{H}$. This equivalence can be shown via a minimax argument (Schapire and Freund, 2012). Albeit being equivalent, the dual assumption is perhaps less intuitive to consider. Indeed, in the context of binary boosting it is often the primal condition that is being used.

There have been various extensions of binary boosting theory to the multiclass setting. Most of these approaches have attempted extending the primal condition. However, it turns out that the original intuition of assuming the learner performs slightly better than random guessing, does not naturally extend to the multiclass case (see Mukherjee and Schapire (2011); Schapire and Freund (2012) and references therein for a detailed discussion).

Though it is still unclear whether one can find a simple condition capturing this notion of "a bit better than random", previous works have proposed a condition that is in some sense optimal. Specifically, Mukherjee and Schapire (2011) have formulated a weak learning assumption, with respect to a carefully tailored loss function, which they prove to be both necessary and sufficient for boosting. Interestingly, this condition is exactly equivalent to the immediate generalization of the *dual* condition mentioned above, to the multiclass case. That is, it corresponds to the assumption that the data can be approximated by a large-margin weighted *plurality-vote* over hypotheses from $\mathcal{H}$.

Recently, Brukhim et al. (2021) followed a similar formulation for multiclass boosting, and proved a hardness result for large label spaces $\mathcal{Y}$. They show that a broad family of boosting algorithms must incur a cost which scales polynomially with $|\mathcal{Y}|$. Concretely, this result holds for all boosting algorithms that are *proper*, i.e., they output a predictor that is also a plurality-vote over hypotheses from $\mathcal{H}$. Indeed, most boosting algorithms in the literature output a majority-vote in the binary case, and a plurality-vote in the multiclass case.

Importantly, the hardness result holds even for target functions that can be learned efficiently by other standard methods, such as empirical risk minimization. Although such methods may only utilize constant running time and samples, it turns out that any proper boosting algorithm will require running time that scales with $|\mathcal{Y}|$, which could possibly be infinite. Thus, the task of multiclass learning via proper boosting is inherently hard. Our main overarching question in this work is,

Question. Within the framework studied by Mukherjee and Schapire (2011); Brukhim et al. (2021), can multiclass boosting be made efficient when the label space $\mathcal{Y}$ is large?

We deduce a strong affirmative answer to this question, by giving the first efficient *improper* multiclass boosting algorithm that has sample and oracle-call complexity that are $\underline{\text{independent}}$ of $|\mathcal{Y}|$. Moreover, it has overall running time that is polynomial in the size of its input.

A key component of our improper boosting approach, is the technique of *list learning* (Brukhim et al., 2022; Charikar and Pabbaraju, 2022). In list learning, instead of predicting a single outcome for a given unseen input, the goal is to provide a short list of predictions. Here we use this technique as an intermediate goal of the algorithm, by which effectively reducing the size of the label space. The generalization analysis relies on sample compression arguments which result in efficient bounds on the sample and oracle complexities. Taken together, these techniques yield an efficient multiclass boosting algorithm.

Lastly, we discuss the closure properties of learnability over aggregations of weak hypotheses. In the binary case, VC classes preserve learnability over various complex aggregations of hypotheses from a base class (Alon et al., 2021). A corollary of our main result is that a plurality-vote over a learnable class is also learnable. We then complement it by showing that other simple aggregations over hypotheses from a learnable class do *not* preserve learnability.

1.1. Main results

Our main result is that there is an efficient multiclass boosting algorithm, as stated in the Theorem 1. Our boosting algorithm is given in Section 4 (Algorithm 2).

In this work we follow the boosting framework of Brukhim et al. (2021) which is based on that of Mukherjee and Schapire (2011). It is a natural formulation in which the weak hypotheses are assumed to belong to an "easy-to-learn" base class $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$, for which the weak learner is an agnostic PAC learner. That is, it can approximate the best hypothesis in $\mathcal{H}$. The goal of the overall boosting algorithm is to learn target concepts *outside* the class $\mathcal{H}$, that can be represented as a large-margin plurality-votes over hypotheses from $\mathcal{H}$.

The class $\mathcal{H}$ consists of *simple* hypotheses, in line with the common supposition that weak hypotheses are rules of thumb from an "easy-to-learn" class (Schapire and Freund, 2012; Shalev-Shwartz and Ben-David, 2014). Thus, the weakness of the weak learner is manifested in the simplicity of the hypotheses in the base class. The simplicity of the base class is measured via its DS dimension, an extension of the VC-dimension to the non-binary setting (Brukhim et al., 2022). We also consider a more general setting in which the class $\mathcal{H}$ is possibly a *partial* class, in which the concepts may not necessarily be defined over the entire domain $\mathcal{X}$. This is based on the formulation introduced by Alon et al. (2022) for binary classification.

Theorem 1 *Let $\epsilon, \delta > 0$, and $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ a hypotheses class with DS dimension d. Let $\mathcal{W}$ be an agnostic PAC learner for $\mathcal{H}$. There exists a multiclass boosting algorithm B such that the following holds. For any distribution $\mathcal{D}$ that is realizable by a γ-margin plurality-vote over hypotheses from $\mathcal{H}$, given access to $\mathcal{W}$ and a training set $S \sim \mathcal{D}^m$ with*

$$m = \tilde{O}\left(\frac{d^{3/2} + \ln(1/\delta)}{\gamma^4 \epsilon^2}\right),$$

with probability $1-\delta$ over the sample S, applying B with $T = \tilde{O}\left(\frac{\ln(m)}{\gamma^2}\right)$ calls to $\mathcal{W}$, outputs a predictor $\bar{h}$ such that,

$$\mathbb{P}_{(x,y)\sim\mathcal{D}}\left[\bar{h}(x) \neq y\right] \leq \epsilon.$$

1.2. Related work

Boosting was first studied in the context of binary PAC learning in a line of seminal works which include the celebrated Adaboost algorithm (see e.g. Kearns (1988); Schapire (1990); Freund (1990); Freund and Schapire (1997)). The boosting methodology was later adapted and applied to a wide range of machine learning applications; such as regression tasks (Mason et al., 2000), recommendation systems (Freund et al., 2003), online learning (Beygelzimer et al., 2015; Brukhim and Hazan, 2021; Brukhim et al., 2020) etc.

In particular, there have been various extensions of boosting to the multiclass setting. These works emphasize practical aspects and provide evidence of improved performance in different applications (Zhai et al., 2014; Kuznetsov et al., 2014; Kégl, 2014; Appel and Perona, 2017; Beijbom et al., 2014; Saberian and Vasconcelos, 2011).

As discussed above, our model is derived within the theoretical framework for multiclass boosting studied by Mukherjee and Schapire (2011), and Brukhim et al. (2021). Our approach circumvents the hardness result given by Brukhim et al. (2021) resulting in an efficient, improper, multiclass boosting algorithm.

2. Background

We study boosting algorithms that aim to learn a combination of hypotheses from a fixed base class $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$, where $\mathcal{X}$ is the data domain, and $\mathcal{Y}$ is the set of labels. In particular, we consider boosting as the task of learning a target concept that can be presented as a *plurality-vote* of hypotheses from $\mathcal{H}$ with margin bounded away from 0. A plurality-vote over the hypotheses class is characterized by a distribution λ over $\mathcal{H}$, and the corresponding classifier $\bar{h} = \bar{h}(\lambda)$ is defined by

$$\bar{h}(x) := \arg\max_{\ell \in \mathcal{Y}} \mathbb{P}_{h\sim\lambda}[h(x) = \ell]. \tag{1}$$

We will overload notation and also use $\bar{h}(x, \ell) = \mathbb{P}_{h\sim\lambda}[h(x) = \ell]$.

Weak learning. The boosting algorithm is given oracle access to $\mathcal{W}$, a weak learner that is an agnostic PAC learner for $\mathcal{H}$. The weakness of $\mathcal{W}$ is manifested in the simplicity of $\mathcal{H}$. Concretely, $\mathcal{W}$ is defined as follows; for every distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, and $\epsilon^w, \delta^w > 0$, when given a sample of $m^w(\epsilon^w, \delta^w)$ examples drawn i.i.d from $\mathcal{D}$, $\mathcal{W}$ outputs $h : \mathcal{X} \to \mathcal{Y}$ such that with probability $1 - \delta^w$,

$$\mathbb{P}_{(x,y)\sim\mathcal{D}}[h(x) = y] \geq \sup_{h^* \in \mathcal{H}} \mathbb{P}_{(x,y)\sim\mathcal{D}}[h^*(x) = y] - \epsilon^w. \tag{2}$$

The weak learner $\mathcal{W}$ considered in this work is defined with respect to the standard classification loss, rather than more complex formulations given in previous work on multiclass boosting.

The assumption made by Mukherjee and Schapire (2011) corresponds to the existence of a γ-margin plurality vote that correctly classifies the fixed training set. Later Brukhim et al. (2021) extended this to the notion of a γ-realizable distribution. We follow the same formulation here.

Definition 2 (***γ-realizable sample/distribution***) *Let $\gamma \in (0, 1)$, $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ a hypotheses class. A sample $S \subseteq \mathcal{X} \times \mathcal{Y}$ is γ-realizable with respect to $\mathcal{H}$ if there exists a distribution λ over $\mathcal{H}$ (corresponding to a plurality-vote) such that for every $(x, y) \in S$ and $\ell \neq y$, it holds that,*

$$\mathbb{E}_{h\sim\lambda}\big[\sigma_h(x, y, \ell)\big] \geq \gamma,$$

where $\sigma_h(x, y, \ell) = \mathbb{1}[h(x) = y] - \mathbb{1}[h(x) = \ell]$. A distribution $\mathbf{D}$ over $\mathcal{X} \times \mathcal{Y}$ is γ-realizable with respect to $\mathcal{H}$ if an independent sample from it is γ-realizable with probability 1.[1]

Overall, the end goal of the boosting algorithm is to approximate a target γ-realizable distribution $\mathbf{D}$ with respect to classification loss. The booster is provided with oracle access to a learner for the base class $\mathcal{H}$, as well as $\epsilon^B, \delta^B > 0$ and $m(\epsilon^B, \delta^B)$ examples drawn i.i.d. from $\mathcal{D}$. The booster makes T iterative calls to the learner and outputs a predictor $\bar{h}_T$ such that with probability $1 - \delta^B$,

$$\mathbb{P}_{(x,y)\sim\mathcal{D}}[\bar{h}_T(x) \neq y] \leq \epsilon^B. \tag{3}$$

1. Our results remain valid for a relaxed definition of a γ-realizable distribution for which a γ-realizable samples are only drawn with high probability. However, we consider the above for simplicity of presentation.

2.1. Partial Class

We follow the formulation presented by Alon et al. (2022), extending it from binary-classification to the multiclass setting. The idea is as follows: rather than learning a class of concepts $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$, where each concept $c \in \mathcal{H}$ is a <u>total</u> function $c \colon \mathcal{X} \to \mathcal{Y}$, we consider partial concept classes $\mathcal{H} \subseteq \left(\mathcal{Y} \bigcup \{\star\}\right)^{\mathcal{X}}$, where each concept c is a *<u>partial</u>* function; specifically, if x is such that $c(x) = \star$ then c is *undefined* at x. The *support* of a partial concept $c \colon \mathcal{X} \to \left(\mathcal{Y} \bigcup \{\star\}\right)$ is the set $\mathsf{supp}(c) := c^{-1}(\mathcal{Y}) = \{x \in \mathcal{X} : c(x) \neq \star\}$.

We next present a characterization of the PAC learnable partial concept classes. But first, we should clarify the definition of PAC learning in this context. Let us begin with the noiseless and realizable setting: intuitively, we want realizability to express the premise that the data drawn from the source distribution satisfy the data-dependent assumptions captured by the partial concept class $\mathcal{H}$. This gives rise to the following definition: a distribution $\mathcal{D}$ on $\mathcal{X} \times \mathcal{Y}$ is *realizable by* $\mathcal{H}$ if almost surely (i.e., with probability 1), a sample $S = ((x_i, y_i))_{i=1}^n \sim \mathcal{D}^n$ (for any n) is realizable by some partial concept $h \in \mathcal{H}$: that is, $\{x_i\}_{i=1}^n \subseteq \text{supp}(h)$, and $h(x_i) = y_i$ for all $i \leq n$.

Definition 3 (PAC Learnability) *A partial concept class $\mathcal{H}$ is* PAC learnable *if, for every $\epsilon, \delta \in (0,1)$, there exists a finite $m := m(\epsilon, \delta) \in \mathbb{N}$ and a learning algorithm $\mathcal{A}$ such that, for every distribution $\mathcal{D}$ on $\mathcal{X} \times \mathcal{Y}$ realizable w.r.t. $\mathcal{H}$, for $S \sim \mathcal{D}^m$, with probability at least $1 - \delta$,*

$$\mathbb{P}_{(x,y)\sim\mathcal{D}}[\mathcal{A}(S)(x) \neq y] \leq \epsilon.$$

Note that prediction loss of a partial concept h with respect to a distribution $\mathcal{D}$ on $\mathcal{X} \times \mathcal{Y}$, is measured via classification error. To be clear, this means we *always* count the case $h(x) = \star$ as a prediction mistake.

Lastly, we remark that throughout this work we consider the general case of partial concept classes. That is, our weak learning assumption, given in Equation (2), as well as the distributional assumption on γ-realizability, are assumed to hold with respect to partial concept classes. Thus, our main results of a boosting algorithm and generalization guarantees hold in this general setting.

2.2. List PAC Learning

A key component of our Boosting algorithm is based on the concept of List PAC learning Brukhim et al. (2022). It is easily explained as an analogy to standard PAC learning. In the PAC setting, the goal is to provide a single prediction on an unseen data point.

In the list learning setting, the goal is instead to provide a short list of predictions. We extend the formulation given by Brukhim et al. (2022), which applied to total classes, to the partial classes as well. The goal of list PAC learning is to return good lists that will generalize to unseen data. We formally define it next.

Definition 4 (p-list) *A list of size $p \in \mathbb{N}$ is a function $\mu : \mathcal{X} \to \{Y \subseteq \mathcal{Y} : |Y| \leq p\}$.*

We will abuse notation and denote $\mu(x)$ at index $\ell \in Y$ by $\mu(x, \ell)$. List PAC learning is the following natural version of standard PAC learning.

Definition 5 (List PAC Learner) *An algorithm $\mathcal{A}$ with sample size n and list size p is a* list PAC learner *for $\epsilon, \delta > 0$ and a* partial *concept class $\mathcal{H} \subseteq \left(\mathcal{Y} \bigcup \{\star\}\right)^{\mathcal{X}}$ if for every $\mathcal{H}$-realizable distribution*

$\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, it holds that

$$\mathbb{P}_{S\sim\mathcal{D}^n}\Big[\mathbb{P}_{(x,y)\sim\mathcal{D}}\big[y \notin \mu_S(x)\big] > \epsilon\Big] \leq \delta,$$

where $\mu_S = A(S)$ is always a p-list.

We define the notion of list realizability, which will be useful in our analysis of the main result, given in Section 5.

Definition 6 (List Realizability) *A sample $S \in (\mathcal{X} \times \mathcal{Y})^n$ is* realizable *by the list μ if $y \in \mu(x)$ for every (x, y) in S. A distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ is* realizable *by μ if for every $m \in \mathbb{N}$, a random sample $S \sim \mathcal{D}^m$ is realizable by μ with probability* 1.

This definition captures the ideal scenario that we have a list that completely captures the entire unknown distribution $\mathcal{D}$. This idealization is a useful sub-goal that we will tackle in the next section.

3. Weak Learner to List Learner

The starting point of our improper boosting algorithm, is a list learning procedure. In the standard PAC setting, the goal is to provide a single prediction on an unseen data point. In list PAC learning, the goal is to provide a short list of predictions. In the following section, we describe a simple reduction of list learning to weak learning.

The following lemma describes a simple guarantee satisfied by the a weak learner, that is later used to obtain a list learner.

Lemma 7 *Let $\mathcal{D}$ denote a γ-realizable distribution with respect to $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$. Then,*

$$\sup_{h^*\in\mathcal{H}} \mathbb{P}_{(x,y)\sim\mathcal{D}}[h^*(x) = y] \geq \gamma.$$

Note that by the weak learning guarantee, we then get the following guarantee. When given $m^w(\gamma/2, \delta)$ i.i.d samples from a γ-realizable distribution $\mathcal{D}$, the learner $\mathcal{W}$ outputs $h \in \mathcal{H}$ such that with probability at least $1 - \delta$,

$$\mathbb{P}_{(x,y)\sim\mathcal{D}}[h(x) = y] \geq \sup_{h^*\in\mathcal{H}} \mathbb{P}_{(x,y)\sim\mathcal{D}}[h^*(x) = y] - \gamma/2 \geq \gamma/2, \tag{4}$$

where the last inequality follows from Lemma 7. Since this gives rise to a guarantee which resembles the standard weak learning condition, it is tempting to try to improve the error by directly applying boosting. But standard boosting turns out to be useless in this context.

The traditional assumption for boosting in the binary setting requires error below $\frac{1}{2}$. The above guarantee is too weak and does not meet the minimal requirements for classical boosting.

However, this guarantee is in fact sufficient for list learning, as we prove next. Denote by $\bar{\mathcal{H}}_\gamma$ the (possibly partial) concept class of all γ-margin plurality votes. The following Lemma shows that a weak learner for $\mathcal{H}$ is also a list learner for $\bar{\mathcal{H}}_\gamma$.

Lemma 8 (Weak Learning implies List Learning) *Let $\mathcal{W}$ be an agnostic PAC learner for $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$, and $\delta > 0$. Then, when given S a γ-realizable sample of size $n \geq m^w(\frac{\gamma}{2}, \frac{\delta\gamma}{\log(2n)})$, and given oracle access to $\mathcal{W}$, Algorithm 1 outputs a list μ_S of size $p = \lceil \log(2n)/\gamma \rceil$, such that with probability at least $1 - \delta$, for all $(x, y) \in S$ it holds that $y \in \mu_S(x)$.*

Algorithm 1 List learner $\mathcal{A}$ for $\bar{\mathcal{H}}_\gamma$

Input: Data $S \in (\mathcal{X} \times \mathcal{Y})^n$, oracle access to $\mathcal{W}$ learner for $\mathcal{H}$.
Output: A p-list μ_S.

```
1: Set $S_1 := S$.
2: for $i = 1, ..., p$ do
3:     Let $\mathcal{U}_i$ denote the uniform distribution over $S_i$.
4:     Let $h_i$ denote the hypothesis output of $\mathcal{W}$ on input sample $S_i' \sim \mathcal{U}_i^{m_w}$.
5:     Set $S_{i+1}$ to be all the points in $S_i$ which $h_i$ predicts incorrectly.
6: end for
7: Return the list defined by
       $\mu_S(x) = \{h_1(x), \ldots, h_p(x)\}$.
```

Proof Observe that in every round $i \in [p]$, Algorithm 1 calls $\mathcal{W}$ with a γ-realizable sample. Recall that by definition, there is a plurality-vote $\bar{h} \in \bar{\mathcal{H}}_\gamma$ such that for each $(x, y) \in S_i \subseteq S$, and $\ell \neq y$, it holds that $\bar{h}(x, y) \geq \bar{h}(x, \ell) + \gamma$. Thus, $\mathcal{U}_i$ is a γ-realizable distribution. We get that for all $i \in [p]$ simultaneously, with probability at least $1 - \delta^w \cdot p = 1 - \delta$,

$$\mathbb{P}_{(x,y)\sim\mathcal{U}_i}[h_i(x) = y] \geq \gamma/2, \tag{5}$$

where this follows from Lemma 7 and Equation 4. Specifically, this implies that $|S_p| \leq (1-\gamma)^p n < e^{-\gamma p} n < 1$. Hence, with probability $1 - \delta$ the output of Algorithm 1 has zero error over S. ■

4. Multiclass Boosting

The high-level idea of our boosting technique is to use the p-list to reduce the unbounded $\mathcal{Y}$ label-set to a label-set of size p. The algorithm follows similarly to previous multiclass boosting techniques, by repeated calls to the weak learner with modified weights over the training data, based on the empirical error of the previous weak hypothesis. However, here this is all done on a converted label space, mapping labels from $\mathcal{Y}$ to $[p]$. The output of the boosting algorithm is both a plurality-vote, as well as the list mapping.

The following lemma is the first step in the analysis of the performance of Algorithm 2. It determines that with sufficiently many samples and oracle calls, it will return a classifier that correctly fits with the training data with high probability.

Lemma 9 *If a training set S is γ-realizable with respect to $\mathcal{H}$, then by applying Algorithm 2 with $m^w \geq m^w(\frac{\gamma}{p}, \frac{\delta}{T})$ in each round $t \leq T$, with $T = \frac{8(\log(m \cdot p))}{\gamma^2}$ and p as in Algorithm 1, we get that with probability at least $1 - \delta$, it outputs $\bar{h}$ that is consistent with S.*

Algorithm 2 Multiclass Adaboost

```
Input: Data $S \in (\mathcal{X} \times \mathcal{Y})^m$, Oracle access to $\mathcal{W}$ learner for $\mathcal{H}$.
Output: A predictor $\bar{h} : \mathcal{X} \mapsto \mathcal{Y}$.

1: Run Algorithm 1 over sample $S$ to obtain $p$-list $\mu$.
2: Initialize: $\tilde{\mathcal{D}}_1(i, \ell) = \frac{1}{m(p-1)}$ if $\mu(x_i, \ell) \neq y_i$, else set to 0, for all $i \in [m], \ell \in [p]$.
3: for $t = 1, \dots, T$ do
4:     Decompose $\tilde{\mathcal{D}}_t$ into $u_t \in \Delta_m, v_t^1, ..., v_t^m \in \Delta_p$, such that for all $i \in [m], \ell \in [p]$,
           $$\tilde{\mathcal{D}}_t(i, \ell) = u_t(i) v_t^i(\ell).$$
5:     Define a distribution $\mathcal{D}_t \in \Delta_{m \times p}$,
           $$\mathcal{D}_t(i, \ell) = \begin{cases} u_t(i) \cdot \frac{2}{p}, & \text{if } \mu(x_i, \ell) = y_i, \\ u_t(i) \cdot \frac{1 - v_t^i(\ell)}{p}, & \text{otherwise.} \end{cases}$$
6:     Draw a sample set from $\mathcal{D}_t^{m_w}$, converting each pair $(i, \ell)$ to $\big(x_i, \mu(x_i, \ell)\big)$.
7:     Let $h_t$ denote the hypothesis output of $\mathcal{W}$ on the converted sample.
8:     Denote $y_i^\mu := \ell$ for $\mu(x_i, \ell) = y_i$, and $h_t(x_i)^\mu$ similarly, or set to $\star$ if $h_t(x_i) \notin \mu(x_i)$.
9:     Compute the empirical edge,
           $$\gamma_t := \sum_{\substack{i \in [m] \\ h_t(x_i)^\mu \neq \star}} u_t(i) \Big( \mathbb{1}\big[h_t(x_i)^\mu = y_i^\mu\big] - \mathbb{P}_{\ell \sim v_t^i}\big[h_t(x_i)^\mu = \ell\big] \Big).$$
10:    Set $\alpha_t = \frac{1}{2} \ln \left( \frac{1+\gamma_t}{1-\gamma_t} \right)$. Update, for all $i \in [m], \ell \in [p]$:
           $$\tilde{\mathcal{D}}_{t+1}(i, \ell) = \frac{\tilde{\mathcal{D}}_t(i, \ell)}{Z_t} \times \begin{cases} e^{-\alpha_t}, & \text{if } h_t(x_i)^\mu = y_i^\mu, \\ e^{\alpha_t}, & \text{if } h_t(x_i)^\mu = \ell, \\ 1, & \text{otherwise.} \end{cases}$$
           $$= \frac{\tilde{\mathcal{D}}_t(i, \ell) \exp(-\alpha_t \cdot \sigma_{h_t^\mu}(x_i, y_i^\mu, \ell))}{Z_t},$$
       where $Z_t$ is a normalization factor (chosen so that $\tilde{\mathcal{D}}_{t+1}$ will be a distribution).
11: end for
12: Return the weighted plurality-vote, restricted to the list output:
           $$\bar{h}(x) := \mu\Big(x\ ,\ \underset{\ell \in [p]}{\text{argmax}} \sum_{t=1}^{T} \alpha_t \cdot \mathbb{1}[h_t^\mu(x) = \ell]\Big).$$
```

4.1. Adaptivity

We briefly discuss the adaptivity properties of Algorithm 2. Boosting algorithms typically do not assume knowing the value of the margin γ and are adapted to it on the fly, as in the well-known Adaboost algorithm Schapire and Freund (2012).

Our boosting framework requires feeding the approximate weak learner (specified in Equation (2)) with $m^w := m^w(\epsilon^w, \delta^w)$ examples, where parameters ϵ^w, δ^w are functions of γ. If m^w is being given as a parameter, then Algorithm 2 is indeed adaptive, and does not require knowing the true γ. Otherwise, a value of γ needs to be estimated in order to set m^w. This can be resolved by a simple procedure which only increases the overall runtime by a logarithmic factor of $O(\ln(1/\gamma))$. For details see Appendix B.2.

5. Generalization bound

In this section we give our main result, formally stated in Theorem 1, which bounds the generalization error of the boosting algorithm, given in Algorithm 2.

The overall bound on the sample complexity of the boosting algorithm can be stated as a function of the following terms: error parameters ϵ, δ, margin value γ, and the sample complexity of the weak learner m^w, defined in Equation (2). Alternatively, we can bound m^w via a complexity measure of the class $\mathcal{H}$, called the *DS dimension*.

The DS dimension was originally introduced by Daniely and Shalev-Shwartz (2014). Here we follow the formulation given in Brukhim et al. (2022), and so we first introduce the notion of *pseudo-cubes*.

Definition 10 (Pseudo-cube) *A class $\mathcal{H} \subseteq \mathcal{Y}^d$ is called a* pseudo-cube *of dimension d if it is non-empty, finite and for every $h \in \mathcal{H}$ and $i \in [d]$, there is an i-neighbor $g \in \mathcal{H}$ of h (i.e., $g(i) \neq h(i)$ and $g(j) = h(j)$ for all $j \neq i$).*

When $\mathcal{Y} = \{0, 1\}$, the two notions "Boolean cube" and "pseudo-cube" coincide: The Boolean cube $\{0, 1\}^d$ is of course a pseudo-cube. Conversely, every pseudo-cube $\mathcal{H} \subseteq \{0, 1\}^d$ is the entire Boolean cube $\mathcal{H} = \{0, 1\}^d$. When $|\mathcal{Y}| > 2$, the two notions do not longer coincide. Every copy of the Boolean cube is a pseudo-cube, but there are pseudo-cubes that are not Boolean cubes (see Brukhim et al. (2022) for further details). We are now ready to define the DS dimension.

Definition 11 (DS dimension) *We say that $S \in \mathcal{X}^n$ is DS-shattered by $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ if $\mathcal{H}|_S$ contains an n-dimensional pseudo-cube. The DS dimension $d_{DS}(\mathcal{H})$ is the maximum size of a DS-shattered sequence.*

The DS dimension of a hypothesis class, $d_{DS}(\mathcal{H})$, characterizes the sample complexity required for PAC-learning the class $\mathcal{H}$. Specifically, by Brukhim et al. (2022) in order to agnostically-PAC learn $\mathcal{H}$ it is sufficient to use a sample complexity of size at most,

$$m_{\mathcal{H}}(\epsilon, \delta) = \tilde{O}\left(\frac{d_{DS}(\mathcal{H})^{3/2} + \ln(1/\delta)}{\epsilon^2}\right). \tag{6}$$

We note that although Brukhim et al. (2022) do not define the DS dimension for a partial class $\mathcal{H}$, the definition and their results are directly applicable for the more general setting as well.

We can now give a formal statement of our generalization result for Algorithm 2. This is restating the result given in Theorem 1.

Theorem 12 (Boosting generalizes) *Let $\epsilon, \delta > 0$, $\mathcal{D}$ a distribution that is γ-realizable by a class $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$, and $\mathcal{W}$ a learner for $\mathcal{H}$. Let $S \sim \mathcal{D}^m$ be a training set for*

$$m = \tilde{O}\left(\frac{m^w}{\gamma^2 \epsilon^2} + \frac{\ln(1/\delta)}{\epsilon^2}\right) = \tilde{O}\left(\frac{d^{3/2} + \ln(1/\delta)}{\gamma^4 \epsilon^2}\right),$$

where $m^w = O\left(m^w(\frac{\gamma^2}{\ln(m)}, \frac{\delta\gamma^3}{\ln(m)^2})\right)$, and $d = d_{DS}(\mathcal{H})$. Then, with probability $1-\delta$ over the sample S, applying Algorithm 2 with $T = \tilde{O}\left(\frac{\ln(m)}{\gamma^2}\right)$, outputs a predictor $\bar{h}$ such that,

$$\mathbb{P}_{(x,y)\sim\mathcal{D}}\left[\bar{h}(x) \neq y\right] \leq \epsilon.$$

The proof of Theorem 12 will occupy the remainder of this section.

5.1. Plurality-vote dimension

We start by upper bounding the complexity of classes of plurality votes. Specifically, here we are interested in a complexity measures called the *Natarajan dimension*. The following result, which bounds this quantity, is used in the analysis of the overall generalization bound, given in Theorem 12.

The Natarajan dimension was defined by Natarajan (1989), as follows.

Definition 13 (Natarajan dimension) *Let $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ a hypothesis class and let $S \subseteq \mathcal{X}$. We say that $\mathcal{H}$* N-shatters *S if $\exists\, \pi_1, \pi_2 : S \to \mathcal{Y}$ such that $\forall x \in S,\ \pi_1(x) \neq \pi_2(x)$, and for every $S' \subseteq S$ there is $h \in \mathcal{H}$ such that,*

$$\forall x \in S',\ h(x) = \pi_1(x),\ \textit{and}\ \forall x \in S \setminus S',\ h(x) = \pi_2(x).$$

The Natarajan dimension *$d_N(\mathcal{H})$, is the maximal cardinality of a set that is N-shattered by $\mathcal{H}$.*

The Natarajan dimension coincides with the VC-dimension for $|\mathcal{Y}| = 2$. The Natarajan dimension of a hypothesis class, $d_N(\mathcal{H})$, characterizes the sample complexity required for PAC-learning the class $\mathcal{H}$, up to a multiplicative factor of $O(\log(|\mathcal{Y}|))$. Specifically, by Ben-David et al. (1995); Daniely et al. (2015) in order to agnostically-PAC learn $\mathcal{H}$ it is sufficient to use a sample complexity of size at most,

$$m_{\mathcal{H}}(\epsilon, \delta) = O\left(\frac{d_N(\mathcal{H}) \log(|\mathcal{Y}|) + \ln(1/\delta)}{\epsilon^2}\right). \tag{7}$$

The plurality-vote class of interest is defined in the following lemma.

Lemma 14 (Natarajan of plurality-votes) *Fix $T, p > 0$, a set of hypotheses $h_1, \ldots, h_T \in \mathcal{H}$, and μ a p-list. Set $h_t^\mu(x) = \ell$ such that $\mu(x, \ell) = h_t(x)$, or set to 0 if $h_t(x) \notin \mu(x)$. For any $\alpha \in \mathbb{R}^T$, define,*

$$f_\alpha(x) = \underset{\ell \in [p]}{\textit{argmax}} \sum_{t=1}^{T} \alpha_t \cdot \mathbb{1}[h_t^\mu(x) = \ell], \tag{8}$$

and let $\mathcal{F} = \{f_\alpha : \alpha \in \mathbb{R}^T\}$. Then, it holds that $d_N(\mathcal{F}) \leq T$.

5.2. Hybrid compression scheme

The Boosting algorithm given in this work is best thought of as a *sample compression scheme* (Littlestone and Warmuth, 1986). A sample compression scheme (Definition 15) is an abstraction of a common property to many learning algorithms. It can be viewed as a two-party protocol between a *compresser* and a *reconstructor*. Both players know the underlying concept class $\mathcal{H}$. The compresser gets as input an $\mathcal{H}$-realizable sample S. The compresser picks a small subsample S' of S and sends it to the reconstructor. The reconstructor outputs an hypothesis h. The correctness criteria is that h needs to correctly classify *all* examples in the input sample S. We formally define it next.

Definition 15 (Sample Compression Scheme (Littlestone and Warmuth, 1986)) *Let $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ and let $r \leq n$ be integers. An $n \to r$* sample compression scheme *consists of a* reconstruction *function*

$$\rho : (\mathcal{X} \times \mathcal{Y})^r \to \mathcal{Y}^{\mathcal{X}}$$

such that for every $\mathcal{H}$-realizable $S \in (\mathcal{X} \times \mathcal{Y})^n$, there exists $S' \subseteq S$ of size r, such that for all $(x, y) \in S$ we have $h(x) = y$, where $h = \rho(S')$.

The main goal of this section is to establish that the output of the improper boosting technique given in Algorithm 2 is a compression scheme. Classical sample compression algorithms typically boil down to a simple one-shot encoding scheme, as given Definition 15. However, our compression scheme is more involved and is comprised of three main components: (1) a "list"-compression, (2) standard compression, and (3) a "hybrid" compression, which relies both on a standard compression argument along with classic generalization theory via the Natarajan dimension. To accommodate this mechanism, we first give the following variant of a sample compression scheme, for a list function Brukhim et al. (2022).

Definition 16 (List Sample Compression Scheme) *An $n \to r$* list sample compression scheme with menu size *p consists of a* reconstruction function

$$\rho : (\mathcal{X} \times \mathcal{Y})^r \to \{Y \subseteq \mathcal{Y} : |Y| \leq p\}^{\mathcal{X}}$$

such that for every $\mathcal{H}$-realizable $S \in (\mathcal{X} \times \mathcal{Y})^n$, there exists $S' \subseteq S$ of size r, such that for all $(x, y) \in S$ we have $y \in \mu(x)$, where $\mu = \rho(S')$.

Next, we will require the use of a *hybrid* sample compression scheme (see (Schapire and Freund, 2012), section 4.2.2), generalized to the multiclass setting, as given in the next Theorem.

Theorem 17 (Hybrid Compression Scheme) *Let $r, \delta > 0$. Assume there is a mapping ψ such that for any set $S' \subset (\mathcal{X} \times \mathcal{Y})^r$, it outputs a class of hypotheses $\psi(S') = \mathcal{F} \subseteq \mathcal{Y}^{\mathcal{X}}$ with Natarajan dimension d. Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \mathcal{Y}$ and $S \sim \mathcal{D}^m$ for $m \geq r + d$. Then, for any $S' \subsetneq S$ of size r, and the corresponding class $\mathcal{F} = \psi(S')$, with probability at least $1 - \delta$, any $f \in \mathcal{F}$ that is consistent with S satisfies,*

$$\mathbb{P}_{(x,y)\sim\mathcal{D}}[f(x) \neq y] \leq O\left(\sqrt{\frac{d \ln(|\mathcal{Y}|) + \ln(m^r/\delta)}{m - r}}\right). \tag{9}$$

We are now ready to prove the main result, given in Theorem 12. The next paragraph highlights the assumptions that were made, followed by the proof of the theorem.

Assumptions Note that in Algorithm 2, boosting by resampling is employed. In other words, we assume on each round t that the weak learner is trained on an *unweighted* sample selected at random by resampling with replacement from the entire training set according to the current distribution $\mathcal{D}_t$. We further assume for simplicity that the learning algorithm $\mathcal{W}$ does not employ randomization, so that it can be regarded as a fixed, deterministic mapping from a sequence of m^w unweighted examples to a hypothesis h. Under this assumption, any hypothesis produced by the learner $\mathcal{W}$ can be represented rather trivially by the very sequence of m^w examples on which it was trained. We note that our results remain valid for a randomized learner as well, yet we assume the above for simplicity.

Proof [Proof of Theorem 12] The proof is given via a sample compression scheme, demonstrating that the final predictor $\bar{h}$ can be represented using a small number of training examples. Specifically, we construct the sample compression scheme in 3 steps: (1) a "list" sample compression (as in Definition 16), which corresponds to the construction of μ, (2) a standard sample compression scheme, which corresponds to the choice of $h_1, ..., h_T$, and (3) a hybrid sample compression, to account for the weights $\alpha_1, ..., \alpha_T$ generated by Algorithm 2.

First, recall the assumption that $\mathcal{W}$ is a deterministic mapping from examples to hypotheses. Therefore, any weak hypothesis produced by the weak learner can be represented simply by the sequence of m^w examples on which it was trained.

Then, observe that step (1) can be shown by Lemma 8. In particular, it implies that for any sample S, there exists a subset $S' \subsetneq S$ of size at most $r_1 = m_1^w \cdot p$, where $m_1^w = m^w(\frac{\gamma}{2}, \frac{\delta}{2pT})$ and $p = \lceil \log(2m)/\gamma \rceil$ such that the following holds: There are $h'_1, ..., h'_p \in \mathcal{H}$, that comprise the list μ, where each h'_i can be represented by m_1^w examples in S'. Moreover, for all $(x, y) \in S$, it holds that $y \in \mu(x)$.

Next, for step (2) notice that the T weak hypotheses $h_1, ..., h_T$ generated in Algorithm 2 can be represented by a sequence of $r_2 = Tm_2^w$ training examples, where $m_2^w = m^w(\frac{\gamma}{2p}, \frac{\delta}{2pT})$.

Lastly, we obtain (3) by combining Theorem 17, with Lemma 14, and Lemma 9. Specifically, consider the class $\mathcal{F}$ defined in Equation (8), for the fixed hypotheses $h_1, ..., h_T$ and for μ, produced by Algorithm 2. By the above steps (1)+(2), we have that the class $\mathcal{F}$ is determined by at most $r = r_1 + r_2$ examples. Moreover, by Lemma 9, since $T = \tilde{O}\left(\frac{\ln(m)}{\gamma^2}\right)$, with probability at least $1 - \delta/2$ Algorithm 2 outputs $\bar{h}$ that is consistent with the entire sample S.

Observe that $\bar{h}$ corresponds to $f \in \mathcal{F}$ that is consistent with a *converted* sample. Specifically, for any sample U of elements in $\mathcal{X} \times \mathcal{Y}$, we define U^μ a converted sample, containing elements in $\mathcal{X} \times [p+1]$, as follows: any $(x, y) \in U$, corresponds to $(x, \ell) \in U^\mu$ such that $\mu(x, \ell) = y$, and otherwise $(x, p+1) \in U^\mu$. We similarly denote $\mathcal{D}^\mu$ as the converted variant of a distribution $\mathcal{D}$. Then,

$$\mathbb{P}_{(x,y)\sim\mathcal{D}}[\bar{h}(x) \neq y] \leq \mathbb{P}_{(x,\ell)\sim\mathcal{D}^\mu}[f(x) \neq \ell] \leq O\left(\sqrt{\frac{T\ln(p) + \ln(m^r/\delta)}{m-r}}\right), \tag{10}$$

where the first inequality follows from the definition of $\bar{h}$ in Algorithm 2, and last inequality follows from Theorem 17 (with respect to class $\mathcal{F}$ and distribution $\mathcal{D}^\mu$), and using the bound on the Natarajan dimension given in Lemma 14. To bound the sample complexity of the weak learner m^w in terms of the DS dimension of $\mathcal{H}$, we plug-in Equation (7) to class $\mathcal{H}$ for which $\mathcal{W}$ is an agnostic PAC learner, with the corresponding parameters to m^w, i.e., $\epsilon^w = \gamma/(2p)$ and $\delta^w = \delta/(2pT)$. ■

6. Closure Properties

In this section we discuss the closure properties of learnability over aggregations of weak hypotheses. In the binary case, VC classes preserve learnability over various complex aggregations of hypotheses from a $\mathcal{H} \subseteq \{\pm 1\}^{\mathcal{X}}$(Alon et al., 2021). The finiteness of the DS dimension characterizes learnability of classes $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ in the multiclass setting (see Section 5 and Brukhim et al. (2022)). Therefore, we essentially ask - what type of aggregations over hypotheses $h \in \mathcal{H}$ can we construct such that if $\mathcal{H}$ has a finite DS dimension, the resulting class of aggregation must have a finite DS dimension as well? The first answer is a corollary of our main result, and pertains to plurality vote aggregations.

Corollary 18 *Let $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ be a hypotheses class with $d = d_{DS}(\mathcal{H}) < \infty$. Let $\gamma > 0$, and denote by $\bar{\mathcal{H}}_\gamma$ the (possibly partial) class of all γ-margin plurality votes over concepts from $\mathcal{H}$. Then, it holds that $d_{DS}(\bar{\mathcal{H}}_\gamma) < \infty$.*

The corollary above is a direct implication of our main result given in Theorem 1. We complement it by showing that other simple aggregations over hypotheses from a learnable class do not preserve learnability, unlike in the binary setting. Specifically, we show this is not the case for an aggregation of dot products over two hypotheses from the base class.

Proposition 19 *Let $\mathcal{Y} \subseteq \{0,1\}^{\mathbb{N}}$. There is a class $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ with $d = d_{DS}(\mathcal{H}) < \infty$ such that for*

$$\mathcal{H}_+ = \left\{ h_+ : x \mapsto h_0(x) \cdot h_1(x) \;\middle|\; h_0, h_1 \in \mathcal{H} \right\},$$

the class of dot products over two hypotheses from $\mathcal{H}$, it holds that $d_{DS}(\mathcal{H}_+) = \infty$.

We note that it can be similarly shown that other simple aggregations over hypotheses from a base class $\mathcal{H}$ such as sums, as well as for the *dual* class. Interestingly, as demonstrated by Corollary 18, this is not the case for plurality-vote aggregations.

Acknowledgments

Shay Moran is a Robert J. Shillman Fellow; he acknowledges support by ISF grant 1225/20, by BSF grant 2018385, by an Azrieli Faculty Fellowship, by Israel PBC-VATAT, by the Technion Center for Machine Learning and Intelligent Systems (MLIS), and by the the European Union (ERC, GENERALIZATION, 101039692). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

References

Noga Alon, Alon Gonen, Elad Hazan, and Shay Moran. Boosting simple learners. *STOC*, 2021.

Noga Alon, Steve Hanneke, Ron Holzman, and Shay Moran. A theory of pac learnability of partial concept classes. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 658–671. IEEE, 2022.

Ron Appel and Pietro Perona. A simple multi-class boosting framework with theoretical guarantees and empirical proficiency. In *International Conference on Machine Learning*, pages 186–194. PMLR, 2017.

Oscar Beijbom, Mohammad Saberian, David Kriegman, and Nuno Vasconcelos. Guess-averse loss functions for cost-sensitive multiclass boosting. In *International Conference on Machine Learning*, pages 586–594. PMLR, 2014.

Shai Ben-David, Nicolo Cesabianchi, David Haussler, and Philip M Long. Characterizations of learnability for classes of $\{0,...,n\}$-valued functions. *Journal of Computer and System Sciences*, 50(1):74–86, 1995.

Alina Beygelzimer, Satyen Kale, and Haipeng Luo. Optimal and adaptive algorithms for online boosting. In *International Conference on Machine Learning*, pages 2323–2331, 2015.

Nataly Brukhim and Elad Hazan. Online boosting with bandit feedback. In *Algorithmic Learning Theory*, pages 397–420. PMLR, 2021.

Nataly Brukhim, Xinyi Chen, Elad Hazan, and Shay Moran. Online agnostic boosting via regret minimization. *Advances in Neural Information Processing Systems*, 33:644–654, 2020.

Nataly Brukhim, Elad Hazan, Shay Moran, and Robert E. Schapire. Multiclass boosting and the cost of weak learning. In *NIPS*, 2021.

Nataly Brukhim, Daniel Carmon, Irit Dinur, Shay Moran, and Amir Yehudayoff. A characterization of multiclass learnability. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 943–955. IEEE, 2022.

Moses Charikar and Chirag Pabbaraju. A characterization of list learnability. *arXiv preprint arXiv:2211.04956*, 2022.

Amit Daniely and Shai Shalev-Shwartz. Optimal learners for multiclass problems. In *COLT*, pages 287–316, 2014.

Amit Daniely, Sivan Sabato, Shai Ben-David, and Shai Shalev-Shwartz. Multiclass learnability and the ERM principle. *The Journal of Machine Learning Research*, 16:2377–2404, 2015.

Yoav Freund. Boosting a weak learning algorithm by majority. In Mark A. Fulk and John Case, editors, *Proceedings of the Third Annual Workshop on Computational Learning Theory, COLT 1990, University of Rochester, Rochester, NY, USA, August 6-8, 1990*, pages 202–216. Morgan Kaufmann, 1990. URL http://dl.acm.org/citation.cfm?id=92640.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997. doi: 10.1006/jcss.1997.1504. URL https://doi.org/10.1006/jcss.1997.1504.

Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969, December 2003. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=945365.964285.

M. Kearns. Thoughts on hypothesis boosting. Unpublished, December 1988.

Balázs Kégl. The return of adaboost. mh: Multi-class hamming trees. *ICLR*, 2014.

Vitaly Kuznetsov, Mehryar Mohri, and Umar Syed. Multi-class deep boosting. In *Advances in Neural Information Processing Systems*, pages 2501–2509, 2014.

Nick Littlestone and Manfred Warmuth. Relating data compression and learnability. *Unpublished manuscript*, 1986.

Llew Mason, Jonathan Baxter, Peter L Bartlett, and Marcus R Frean. Boosting algorithms as gradient descent. In *Advances in neural information processing systems*, pages 512–518, 2000.

Indraneel Mukherjee and Robert E Schapire. A theory of multiclass boosting. *Journal of Machine Learning Research*, 14:437–497, 2011. ISSN 1532-4435. doi: citeulike-article-id:8467733. URL http://rob.schapire.net/papers/multiboost-journal.pdfhttp://arxiv.org/abs/1108.2989.

Balas K Natarajan. On learning sets and functions. *Machine Learning*, 4(1):67–97, 1989.

Mohammad J Saberian and Nuno Vasconcelos. Multiclass boosting: Theory and algorithms. In *Advances in Neural Information Processing Systems*, pages 2124–2132, 2011.

Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990. doi: 10.1007/BF00116037. URL https://doi.org/10.1007/BF00116037.

Robert E Schapire and Yoav Freund. *Boosting: Foundations and algorithms*. Cambridge university press, 2012.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

Shaodan Zhai, Tian Xia, and Shaojun Wang. A multi-class boosting method with direct optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 273–282, 2014.

Appendix A. Missing proofs of Section 2.2

Proof [Proof of Lemma 7] By the assumption, there exists a distribution $\lambda \in \Delta_{\mathcal{H}}$ such that for any $(x, y) \in \mathcal{X} \times \mathcal{Y}$ in the support of $\mathcal{D}$, and any other label $\ell \neq y$,

$$\mathbb{E}_{h\sim\lambda}\Big[\mathbb{1}[h(x) = y] - \mathbb{1}[h(x) = \ell]\Big] = \mathbb{E}_{h\sim\lambda}\big[\sigma_h(x, y, \ell)\big] \geq \gamma. \tag{11}$$

Thus, for all such (x, y) we have,

$$\mathbb{E}_{h\sim\lambda}\big[\mathbb{1}[h(x) = y]\big] \geq \gamma. \tag{12}$$

By taking expectation with respect to $\mathcal{D}$, and changing order of summations we get,

$$\mathbb{E}_{h\sim\lambda}\Big[\mathbb{P}_{(x,y)\sim\mathcal{D}}\left[h(x) = y\right]\Big] \geq \gamma. \tag{13}$$

This implies that there exists a *particular* $h \in \mathcal{H}$ such that,

$$\mathbb{P}_{(x,y)\sim\mathcal{D}}\left[h(x) = y\right] \geq \gamma, \tag{14}$$

which concludes the proof. ■

Appendix B. Missing proofs of Section 4

Lemma 20 *If a training set S is γ-realizable with respect to $\mathcal{H}$, then by applying Algorithm 2 with $m^w = m^w(\gamma/p, \delta/T)$ in each round t, we get that with probability at least $1-\delta$, it yields a sequence of empirical edges $\gamma_1, ..., \gamma_T$ such that for all $t \leq T$, it holds that $\gamma_t \geq \gamma/2$.*

Proof Observe that for any $h \in \mathcal{H}$,

$$\mathbb{P}_{\mathcal{D}_t}[h(x)=y] = \sum_{i=1}^{m} u_t(i)\left(\frac{2}{p}\mathbb{1}[h(x_i)=y_i] + \sum_{\ell \neq y_i} \frac{1-v_t^i(\ell)}{p}\mathbb{1}[h(x_i)=\ell]\right) \tag{15}$$

$$= \frac{2}{p}\mathbb{P}_{\tilde{\mathcal{D}}_t}[h(x)=y] + \frac{1}{p}(1-\mathbb{P}_{\tilde{\mathcal{D}}_t}[h(x)=y]) - \frac{1}{p}\sum_{i=1}^{m}\sum_{\ell\neq y_i}\tilde{\mathcal{D}}_t(i,\ell)\mathbb{1}[h(x_i)=\ell] \tag{16}$$

$$= \frac{1}{p}\mathbb{P}_{\tilde{\mathcal{D}}_t}[h(x)=y] + \frac{1}{p} - \frac{1}{p}\sum_{i=1}^{m}\sum_{\ell\neq y_i}\tilde{\mathcal{D}}_t(i,\ell)\mathbb{1}[h(x_i)=\ell] \tag{17}$$

$$= \frac{1}{p} + \frac{1}{p}\left(\mathbb{P}_{\tilde{\mathcal{D}}_t}[h(x)=y] - \sum_{i=1}^{m}\sum_{\ell\neq y_i}\tilde{\mathcal{D}}_t(i,\ell)\mathbb{1}[h(x_i)=\ell]\right). \tag{18}$$

Then, since for each round t, the empirical distribution $\mathcal{D}_t$ is a γ-realizable distribution, by the weak learning guarantee we have with probability of at least $1-\delta^w \cdot T = 1-\delta$ simultaneously for all t,

$$\mathbb{P}_{\mathcal{D}_t}[h_t(x)=y] \geq \max_{h^*\in\mathcal{H}} \mathbb{P}_{\mathcal{D}_t}[h^*(x)=y] - \epsilon^w. \tag{19}$$

Then, combining with Equation (18) we get,

$$\frac{1}{p} + \frac{\gamma_t}{p} = \frac{1}{p} + \frac{1}{p}\left(\mathbb{P}_{\tilde{\mathcal{D}}_t}[h_t(x)=y] - \sum_{i=1}^{m}\sum_{\ell\neq y_i}\tilde{\mathcal{D}}_t(i,\ell)\mathbb{1}[h(x_i)=\ell]\right) \qquad \text{(definition of } \gamma_t)$$

$$= \mathbb{P}_{\mathcal{D}_t}[h_t(x)=y] \qquad \text{(Equation (18))}$$

$$\geq \max_{h^*\in\mathcal{H}} \mathbb{P}_{\mathcal{D}_t}[h^*(x)=y] - \epsilon^w \qquad \text{(agnostic learning guarantee)}$$

$$= \frac{1}{p} + \frac{1}{p}\max_{h^*\in\mathcal{H}}\left(\mathbb{P}_{\tilde{\mathcal{D}}_t}[h^*(x)=y] - \sum_{i=1}^{m}\sum_{\ell\neq y_i}\tilde{\mathcal{D}}_t(i,\ell)\mathbb{1}[h^*(x_i)=\ell]\right) - \epsilon^w \qquad \text{(Equation (18))}$$

$$\geq \frac{1}{p} + \frac{\gamma}{p} - \epsilon^w. \qquad (\gamma\text{-realizability})$$

Thus, by re-arranging terms and setting $\epsilon^w = \frac{\gamma}{2p}$ we get $\gamma_t \geq \gamma/2$. ■

B.1. Proof of Lemma 9

Proof Throughout the proof, we only consider the high-probability event in which Lemma 20 holds. Let p be as chosen in Algorithm 1. For any $x \in \mathcal{X}, \ell \in [p]$, we define:

$$F(x,\ell) = \sum_{t=1}^{T} \alpha_t \cdot \mathbb{1}[h_t(x) = \ell]. \tag{20}$$

Unraveling the recurrence in Algorithm 2 that defines $\tilde{\mathcal{D}}_{t+1}$ in terms of $\tilde{\mathcal{D}}_t$ gives,

$$\begin{aligned}
\tilde{\mathcal{D}}_{T+1}(i,\ell) &= \tilde{\mathcal{D}}_1(i,\ell) \times \frac{e^{-\alpha_1 \cdot \sigma_{h_1}(x_i, y_i^\mu, \ell)}}{Z_1} \times \ldots \times \frac{e^{-\alpha_T \cdot \sigma_{h_T^\mu}(x_i, y_i^\mu, \ell)}}{Z_T} \\
&= \tilde{\mathcal{D}}_1(i,\ell) \times \frac{e^{-\sum_{t=1}^{T} \alpha_t \cdot \sigma_{h_t^\mu}(x_i, y_i^\mu, \ell)}}{\prod_{t=1}^{T} Z_t} \\
&= \tilde{\mathcal{D}}_1(i,\ell) \times \frac{e^{-\sum_{t=1}^{T} \alpha_t \cdot \left(\mathbb{1}[h_t(x_i)^\mu = y_i^\mu] - \mathbb{1}[h_t(x_i)^\mu = \ell]\right)}}{\prod_{t=1}^{T} Z_t} \\
&= \tilde{\mathcal{D}}_1(i,\ell) \times \frac{e^{-F(x_i, y_i^\mu) + F(x_i, \ell)}}{\prod_{t=1}^{T} Z_t}.
\end{aligned} \tag{21}$$

Next, note that $\bar{h}(x) = \mu\Big(x\ ,\ \text{argmax}_{\ell \in [p]} F(x,\ell)\Big)$. Therefore, for any $i \in [m]$ and $\ell \neq y_i^\mu$, we get that if $\bar{h}(x_i)^\mu = \ell$, then $F(x_i,\ell) \geq F(x_i, y_i^\mu)$. This implies that $e^{F(x_i,\ell) - F(x_i, y_i^\mu)} \geq 1$. Hence, we have, $\mathbb{1}[\bar{h}(x_i)^\mu = \ell] \leq e^{F(x_i,\ell) - F(x_i, y_i^\mu)}$. We can now bound the label-weighted error:

$$\sum_{i=1}^{m} \sum_{\ell \neq y_i^\mu} \tilde{\mathcal{D}}_1(i,\ell) \cdot \mathbb{1}[\bar{h}(x_i)^\mu = \ell] \leq \sum_{i=1}^{m} \sum_{\ell \neq y_i^\mu} \tilde{\mathcal{D}}_1(i,\ell) \cdot e^{F(x_i,\ell) - F(x_i, y_i^\mu)} \tag{22}$$

$$= \sum_{i=1}^{m} \sum_{\ell \neq y_i^\mu} \tilde{\mathcal{D}}_{T+1}(i,\ell) \prod_{t=1}^{T} Z_t \tag{23}$$

$$= \prod_{t=1}^{T} Z_t, \tag{24}$$

where equation (23) uses equation (21), and equation (24) uses the fact that $\tilde{\mathcal{D}}_{T+1}$ is a distribution which sums to 1 (over all examples and their *incorrect* labels). Denote,

$$q_t = \sum_{\substack{i \in [m] \\ h_t(x_i)^\mu \neq \star}} \sum_{\ell \neq y_i^\mu} \tilde{\mathcal{D}}_t(i,\ell) \cdot \mathbb{1}[h_t(x_i)^\mu = \ell],$$

and observe that,

$$\gamma_t = \sum_{i \in [m]} \sum_{\ell \neq y_i^\mu} \tilde{\mathcal{D}}_t(i,\ell) \cdot \mathbb{1}[h_t(x_i)^\mu = y_i^\mu] - q_t.$$

Then, bounding the normalization factor,

$$Z_t = \sum_{i=1}^{m} \sum_{\ell \neq y_i^\mu} \tilde{\mathcal{D}}_t(i,\ell) \cdot e^{-\alpha_t \cdot \sigma_{h_t^\mu}(x_i, y_i^\mu, \ell)} \tag{25}$$

$$= \sum_{i=1}^{m} \sum_{\ell \neq y_i^\mu} \tilde{\mathcal{D}}_t(i,\ell) \cdot \Big(\mathbb{1}[h_t(x_i)^\mu = y_i^\mu] \cdot e^{-\alpha_t} + \mathbb{1}[h_t(x_i)^\mu = \ell] \cdot e^{\alpha_t} + \mathbb{1}[h_t(x_i)^\mu \notin \{y_i^\mu, \ell, \star\}] \Big) \tag{26}$$

$$= (q_t + \gamma_t) \cdot e^{-\alpha_t} + q_t \cdot e^{\alpha_t} + (1 - 2q_t - \gamma_t), \tag{27}$$

where equation (27) simply follows by the definition of q_t and γ_t (step 5 of Algorithm 2). Observe that by plugging our choice of α_t in to equation (27) and re-arranging terms we get,

$$Z_t = 2q_t \Big(\frac{1}{\sqrt{1-\gamma_t^2}} - 1 \Big) + 1 - \gamma_t + \gamma_t \sqrt{\frac{1-\gamma_t}{1+\gamma_t}}. \tag{28}$$

Note that the coefficient of q_t is a positive term for any $\gamma_t \in (0,1)$. Therefore, since $(1 - 2q_t - \gamma_t) \geq 0$, and so $q_t \leq \frac{1}{2} - \frac{\gamma_t}{2}$, we get,

$$Z_t \leq (1-\gamma_t)\Big(\frac{1}{\sqrt{1-\gamma_t^2}} - 1 \Big) + 1 - \gamma_t + \gamma_t \sqrt{\frac{1-\gamma_t}{1+\gamma_t}} \tag{29}$$

$$= \sqrt{\frac{1-\gamma_t}{1+\gamma_t}} + \gamma_t \sqrt{\frac{1-\gamma_t}{1+\gamma_t}} \tag{30}$$

$$= (1+\gamma_t)\sqrt{\frac{1-\gamma_t}{1+\gamma_t}} \tag{31}$$

$$= \sqrt{1-\gamma_t^2}. \tag{32}$$

Lastly, plugging into equation (24) gives,

$$\sum_{i=1}^{m} \sum_{\ell \neq y_i^\mu} \tilde{\mathcal{D}}_1(i,\ell) \cdot \mathbb{1}[\bar{h}(x_i)^\mu = \ell] \leq \prod_{t=1}^{T} \sqrt{1-\gamma_t^2} \tag{33}$$

$$\leq e^{-1/2 \cdot \sum_{t=1}^{T} \gamma_t^2} \tag{34}$$

$$\leq e^{-T\gamma^2/8} \tag{35}$$

$$\leq \frac{1}{m \cdot p}, \tag{36}$$

where equation (34) follows by applying the approximation $1 + r \leq e^r$ for all real r, equation (35) follows by Lemma 20. Equation (36) follows by plugging T. Since $\tilde{\mathcal{D}}_1$ is uniform over all examples and incorrect labels, then if the label-weighted training error of the combined classifier $\bar{h}$, which is always an integer multiple of $1/m(p-1)$, is at most $1/(m \cdot p)$, then the training error must in fact be zero.

■

B.2. Adaptivity

We continue the discussion on the adaptivity properties of Algorithm 2, from Section B.2. Our boosting framework requires feeding the approximate weak learner (specified in Equation (2)) with $m^w := m^w(\epsilon^w, \delta^w)$ examples, where parameters ϵ^w, δ^w are functions of γ. If m^w is being given as a parameter, then Algorithm 2 is indeed adaptive, and does not require knowing the true γ. Otherwise, a value of γ needs to be estimated in order to set m^w. This issue has two occurrences: the first is in Line 4 of Algorithm 1, where the weak learner is used to produce a list, and the second is in Line 6 of Algorithm 2, where the weak learner is used for boosting. In particular, if these occurrences of m^w are parameterized by an estimation of γ that is too large, the algorithm may fail.

This can be resolved by a simple preliminary binary-search-type procedure, in which we guess gamma, and possible halve it based on the observed outcome. Specifically, for Algorithm 1, we can estimate γ by making an initial guess of $\tilde{\gamma}$, and run Algorithm 1 with the parameters specified in Lemma 8, replacing γ with $\tilde{\gamma}$. If after $p = \lceil \log(2m)/\tilde{\gamma} \rceil$, the training data is not realizable by the output list, halve $\tilde{\gamma}$ and return the process. With high probability, after at most $O(\log(1/\gamma))$ such rounds, the procedure is guaranteed to succeed. Note that this does not affect the sample complexity, or the size of the list, and only increases the overall runtime by a logarithmic factor.

The same idea can be applied to Algorithm 2. For each round t, the indication that $\tilde{\gamma}$ should be halved is if the value γ_t defined in Line 9 is not strictly positive. After a positive γ_t has been observed we should halve $\tilde{\gamma}$, and denote the hypothesis returned as h_t (and discard previously returned hypothesis at round t). Then, h_t is guaranteed to have an empirical edge that is at least half of the true edge of the class $\mathcal{H}$ for the distribution $\mathcal{D}_t$. As in the previous case, this does not affect the sample complexity, and only increases the overall runtime by a logarithmic factor of $O(\ln(1/\gamma))$.

Appendix C. Missing proofs of Section 5

C.1. Proof of Theorem 17

Proof Let ϵ be equal to the quantity on the right-hand side of Equation (9). First, we fix indices $i_1, ..., i_r \in [m]$. Then, given the random training set S, we denote $S' = \{(x_{i_1}, y_{i_1}), ..., (x_{i_r}, y_{i_r})\}$. Note that this also defines the class $\mathcal{F}$. Moreover, because the training examples in S are assumed to be independent, the training points *not* in S', i.e., $S'' = S \setminus S'$, are also independent of the class $\mathcal{F}$.

Next, we apply the Fundamental Theorem of Learning Theory for the multiclass case (see Theorem 29.3, Shalev-Shwartz and Ben-David (2014), part 1), regarding $\mathcal{F}$ as the hypothesis space and S'' as a training set of size at least $m - r$ and where we replace δ by δ/m^r. Then, with probability at least $1 - \delta/m^r$, this result implies that,

$$\mathbb{P}_{(x,y)\sim\mathcal{D}}[f(x) \neq y] \leq \mathbb{P}_{S''}[f(x) \neq y] + \epsilon = \epsilon,$$

for every $f \in \mathcal{F}$ that is consistent with S'', and in particular for every $f \in \mathcal{F}$ that is consistent with the entire sample S.

Thus we have argued that for any fixed choice of indices $i_1, ..., i_r \in [m]$, with probability $1 - \delta/m^r$, we have $\mathbb{P}_{(x,y)\sim\mathcal{D}}[f(x) \neq y] \leq \epsilon$ for every consistent $f \in \mathcal{F}$. Therefore, by the union bound, since there are m^r choices for these indices, this holds for all sequences of indices with probability at least $1 - \delta$, which concludes the proof. ■

C.2. Proof of Lemma 14

Proof First we show that $d_N(\mathcal{F}) \leq d_N(\mathcal{F}')$, where $\mathcal{F}'$ is defined as follows,

$$\mathcal{F}' = \left\{ f' : A \in \mathbb{R}^{p\times T} \mapsto \underset{\ell\in[p]}{\text{argmax}}\ A\alpha \;\middle|\; \alpha \in \mathbb{R}^T \right\}. \tag{37}$$

By definition of d_N there exists a set $S \subseteq \mathcal{X}$ of $d := d_N(\mathcal{F})$ points that is N-shattered by $\mathcal{F}$. Let $\pi_1, \pi_2 : S \to \mathcal{Y}$ be the corresponding functions that indicate it.

We will show that there is a set U of elements in $\mathbb{R}^{p\times T}$ that is of size $|S|$ and is N-shattered by $\mathcal{F}'$. For each $x \in S$, define $A^x \in \mathbb{R}^{p\times T}$ such that $A^x_{\ell,t} = \mathbb{1}[h^\mu_t(x) = \ell]$. We now show $\psi_1, \psi_2 : U \to [p]$ that witness the shattering of U. For any $A^x \in U$, set $\psi_1(A^x) := \pi_1(x)$, and similarly set $\psi_2(A^x) = \pi_2(x)$.

To see why U is shattered, consider any $S' \subseteq S$ and a corresponding $f \in \mathcal{F}$, such that for all $x \in S'$, $f(x) = \pi_1(x)$, and for all $x \in S \setminus S'$, $f(x) = \pi_2(x)$. Note that f is parameterized by some $\alpha \in \mathbb{R}^T$.

Then, let $U' \subseteq U$ such that it contains the elements corresponding to $x \in S'$, i.e., $U' = \{A^x : x \in S'\}$. Observe that for all $A \in U'$, we have that $\underset{\ell\in[p]}{\text{argmax}}\ A\alpha = \psi_1(A)$, and for all $A \in U \setminus U'$, we have that $\underset{\ell\in[p]}{\text{argmax}}\ A\alpha = \psi_2(A)$. Thus, we get that $d_N(\mathcal{F}) \leq d_N(\mathcal{F}')$.

Lastly, we apply a result given in Shalev-Shwartz and Ben-David (2014), Theorem 29.7, which states that $d_N(\mathcal{F}') \leq T$. This concludes the proof. ■

Proof [Proof of Proposition 19] Let $\mathcal{Y} = \{s \mid \exists n \in \mathbb{N}.\ s \in \{0,1\}^n\}$. For $\mathcal{X} = \mathbb{N}$, let $\mathcal{H}$ such that for every $n \in \mathbb{N}$, and every n-bit binary string $s \in \{0,1\}^n$, there is $h_0^{(s)} \in \mathcal{H}$ such that for all $i \leq n$, it is defined to be $h_0^{(s)}(i) = s(i) \circ s \circ 0$, where $\circ$ denotes concatenation. Similarly, define $h_1^{(s)} \in \mathcal{H}$ such that $h_1^{(s)}(i) = s(i) \circ \neg\, s \circ 1$. That is, we have that for n-bit binary strings s, and $i \leq n$, it holds that $h_0^{(s)}(i) \cdot h_1^{(s)}(i) = s(i)$.

Observe that the class $\mathcal{H}$ is easily learnable, since for any realizable sample of constant size, we can take any point (x, y) in the sample, remove the first and last bit and construct $h_j^{(s)}$ by its identifying string s in the label, and by the last bit of the label j.

However, we get that $\mathcal{H}_+ \subseteq \{0,1\}^{\mathcal{X}}$ is a binary class so that for every integer $n \in \mathbb{N}$ there is a sequence of examples $x_1, ..., x_n \in \mathcal{X}$ that are shattered by some function in $\mathcal{H}_+$. Therefore, the VC dimension of $\mathcal{H}_+$ is infinite. Since for binary classification the VC and DS dimension coincide, the claim follows. ■