Boosting, Voting Classifiers and Randomized Sample Compression Schemes

Arthur da Cunha DAC@CS.AU.DK and **Kasper Green Larsen** LARSEN@CS.AU.DK
Aarhus University

Martin Ritzert RITZERT@INFORMATIK.UNI-GOETTINGEN.DE
Georg-August Universität Göttingen

Editors: Gautam Kamath and Po-Ling Loh

Abstract

In *Boosting*, we aim to leverage multiple *weak learners* to produce a *strong learner*. At the center of this paradigm lies the concept of building the strong learner as a *voting classifier*, which outputs a weighted majority vote of the weak learners. While many successful Boosting algorithms, such as the iconic AdaBoost, produce voting classifiers, their theoretical performance has long remained sub-optimal: The best known bounds on the number of training examples necessary for a voting classifier to obtain a given accuracy has so far always contained at least two logarithmic factors above what is known to be achievable by general *weak-to-strong* learners. In this work, we break this barrier by proposing a randomized Boosting algorithm that outputs voting classifiers whose generalization error contains a single logarithmic dependency on the sample size. We obtain this result by building a general framework that extends sample compression methods to support randomized learning algorithms based on sub-sampling.

Keywords: Boosting, Voting Classifiers, Generalization Bounds, Sample Compression Schemes

1. Introduction

Boosting is a powerful machine learning primitive that allows improving the performance of a base learning algorithm $\mathcal{A}$ by training a committee/ensemble of classifiers. The classic AdaBoost (Freund and Schapire, 1997) algorithm for binary classification is perhaps the most well-known Boosting algorithm. Given an input domain $\mathcal{X}$ and a set $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ of n labeled samples from $\mathcal{X} \times \{-1, 1\}$, the main idea of AdaBoost is to iteratively invoke $\mathcal{A}$ on reweighed versions of S. Each invocation returns a hypothesis $h_t \colon \mathcal{X} \to \{-1, 1\}$ to be combined into a final *voting classifier* f as $f(x) = \operatorname{sign}(\sum_{t=1}^{T} \alpha_t h_t(x))$ for constants $\alpha_t > 0$. The weights used at iteration t are such that samples (x_i, y_i) that are misclassified by many previous hypotheses h_j with $j < t$ receive a large weight, and correctly classified samples receive smaller weights. This intuitively guides the attention of $\mathcal{A}$ towards samples with which the previous hypotheses struggle. More modern variants of Boosting include the highly practical XGBoost (Chen and Guestrin, 2016) and LightGBM (Ke et al., 2017) implementations of Gradient Boosting (Friedman, 2001). See the survey by Natekin and Knoll (2013) for more on Boosting and its applications.

Weak-to-Strong Learning. Historically, Boosting was invented to address a theoretical question of Kearns (1988); Kearns and Valiant (1994) on weak-to-strong learning. A γ-weak learner $\mathcal{W}$ is a learning algorithm which, when queried with a training set S and a distribution $\mathcal{D}$ over S, returns a hypothesis h with $R_{\mathcal{D}}(h) \leq 1/2 - \gamma$. Here $R_{\mathcal{D}}(h) = \Pr_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}[h(\mathbf{x}) \neq \mathbf{y}]$. An (ε, δ)-strong learner on the other hand, is a learning algorithm such that for any distribution $\mathcal{D}$ over $\mathcal{X} \times \{-1, 1\}$, when given $m(\varepsilon, \delta)$ i.i.d. samples from $\mathcal{D}$, returns with probability at least $1 - \delta$ a hypothesis $f \colon \mathcal{X} \to \{-1, 1\}$ with $R_{\mathcal{D}}(f) \leq \varepsilon$. A strong learner may, thus, achieve arbitrarily high accuracy when given enough samples.

With these definitions, Kearns and Valiant asked whether it is always possible to obtain a strong learner from a weak learner. This was answered affirmatively (Schapire, 1990), and AdaBoost is the prototypical such weak-to-strong learner. A natural question is: Given n samples, what is the smallest $R_{\mathcal{D}}(f)$ achievable for a weak-to-strong learner when given access to a γ-weak learner $\mathcal{W}$? Letting $\mathcal{H}$ denote a hypothesis set such that $\mathcal{W}$ always outputs hypotheses from $\mathcal{H}$, if $\mathcal{H}$ has VC-dimension d, Shalev-Shwartz and Ben-David (2014) showed that with probability greater than $1-\delta$, AdaBoost outputs a voting classifier f with

$$R_{\mathcal{D}}(f) = O\left(\frac{d\ln(n/d)\ln n}{\gamma^2 n} + \frac{\ln(1/\delta)}{n}\right). \tag{1}$$

This bound remains the best known for any weak-to-strong learner that outputs a voting classifier: One which makes predictions by taking a weighted majority vote among a set of base classifiers.

On the lower bound side, Larsen and Ritzert (2022) showed that for any weak-to-strong learner, with constant probability over a set of n training samples, the produced hypothesis f satisfies

$$R_{\mathcal{D}}(f) = \Omega\left(\frac{d}{\gamma^2 n}\right).$$

Note that this holds for all weak-to-strong learners, not just those that output a voting classifier. Furthermore, they complemented the lower bound by a Boosting algorithm achieving an optimal

$$R_{\mathcal{D}}(f) = O\left(\frac{d}{\gamma^2 n} + \frac{\ln(1/\delta)}{n}\right). \tag{2}$$

Thus, at a high level, the sample complexity of weak-to-strong learning is fully understood. However, the algorithm by Larsen and Ritzert is somewhat contrived as the produced hypothesis is a majority-of-majorities and *not* a voting classifier. Concretely, using recent results to simplify their algorithm (Larsen, 2023), Larsen and Ritzert combine classic Bagging by Breiman (1996) with a variant of AdaBoost known as AdaBoost$^*_\nu$ (Rätsch et al., 2005). They thus create multiple subsamples of the training data, train a voting classifier on each, and combine them by taking a majority of their predictions.

Contribution I: A New Voting Classifier. In light of the above, it remains a natural and basic theoretical question whether the optimal weak-to-strong learning sample complexity in Eq. (2) can be achieved by a simple voting classifier.

Our first main contribution is a new Boosting algorithm, shown as Algorithm 1, that produces a voting classifier with an improved generalization error in terms of the sample size n. In the algorithm description, $a > 0$ is a sufficiently large constant. We prove the following sample complexity bound for Algorithm 1:

Theorem 1 *There exists universal constant $C > 0$ for which the following holds. Let $\mathcal{D}$ be an unknown distribution over $\mathcal{X} \times \{-1, 1\}$ and let $\mathbf{S} \sim \mathcal{D}^n$. Then for every $\delta > 0$, it holds with probability at least $1 - \delta$ over $\mathbf{S}$ and the randomness of Algorithm 1 with $\mathbf{S}$, δ, a γ-weak learner $\mathcal{W}$ and $N = n$ as input, that the voting classifier $\mathbf{g} = \operatorname{sign}(\mathbf{f})$ produced satisfies*

$$R_{\mathcal{D}}(\mathbf{g}) \leq C \cdot \min \left\{ \frac{(d + \ln(1/\gamma)) \ln(n/\delta)}{\gamma^4 n}, \frac{d \ln(n/d) \ln n}{\gamma^2 n} + \frac{\ln(1/\delta)}{n} \right\}.$$

While it can reduce to the previous best bounds in some regimes, it is the first voting classifier that can achieve a sample complexity with a single logarithmic dependency on n.

Algorithm 1: Sampled Boosting

Input: Training set $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, γ-weak learner $\mathcal{W}$, failure probability δ, upper bound $N \geq n$.

Result: A voting classifier f.

$\mathcal{D}_1 \leftarrow \left(\frac{1}{n}, \ldots, \frac{1}{n}\right)$

$\alpha \leftarrow \frac{1}{2} \ln \frac{1/2+\gamma/2}{1/2-\gamma/2}$ `// guaranteed instead of empirical error`

$m \leftarrow a \cdot \gamma^{-2}(d + \ln(1/\gamma))$ `// subsample size`

$K \leftarrow 32 \cdot (\gamma^{-2} \ln(N/\delta) + 1)$ `// fixed size of final ensemble`

for $k = 1, \ldots, K$ **do**

  Draw m samples $\mathbf{S}_k \sim \mathbf{D}_k^m$

  Invoke $\mathcal{W}$ on $\mathbf{S}_k$ with the uniform distribution to obtain $\mathbf{h}_k$

  for $i = 1, \ldots, n$ **do** `// standard AdaBoost weight update`

    $\mathbf{D}_{k+1}(i) \leftarrow \mathbf{D}_k(i) \exp(-\alpha y_i \mathbf{h}_k(x_i))$

  $\mathbf{Z}_k \leftarrow \sum_{i=1}^n \mathbf{D}_k(i) \exp(-\alpha y_i \mathbf{h}_k(x_i))$

  $\mathbf{D}_{k+1} \leftarrow \mathbf{D}_{k+1} / \mathbf{Z}_k$

return $\mathbf{f}(x) = \frac{1}{K} \sum_{k=1}^K \mathbf{h}_k(x)$ `// majority vote`

At a high level, our new algorithm creates numerous small sub-samples of the training data and combines classifiers trained on each of them. Proving that this is beneficial requires highly novel analysis techniques. Our second main contribution is thus a new general framework for analyzing randomized learning algorithms that use sub-sampling during training. This method builds on the sample compression framework of Littlestone and Warmuth (1986) and we hope it may prove useful in the future development and analysis of efficient learning algorithms. We introduce this new framework in the following subsection and then discuss the connection between Algorithm 1 and the framework.

1.1. Sample Compression Schemes

Learning and compression have been known to be tightly connected for decades. One of the earliest and clearest connections between the two originates in the work of Littlestone and Warmuth (1986). In essence, they argue that if the hypothesis produced by a learning algorithm can be *compressed* to be fully described as a function of a few training samples, then it generalizes well. We describe this connection further in the following.

Let $\mathcal{X}$ be an input domain and $\mathcal{Y}$ an output domain. A compression scheme (κ, ρ) consists of an encoding map κ that maps any sequence $S \in (\mathcal{X} \times \mathcal{Y})^*$ to a subsequence $\kappa(S)$ of S, and a

reconstruction function $\rho\colon (\mathcal{X} \times \mathcal{Y})^* \to \mathcal{Y}^{\mathcal{X}}$ mapping any $S \in (\mathcal{X} \times \mathcal{Y})^*$ to a function $\rho(S)\colon \mathcal{X} \to \mathcal{Y}$. The compression scheme must satisfy for any S that $\rho(\kappa(S))(x) = y$ for all $(x, y) \in S$. The size of the compression scheme is the supremum over S of $|\kappa(S)|$, for given a given size of S. Notably, some notions of compression schemes forgo this dependency on the sample size, e.g., in Moran and Yehudayoff (2016).

Consider now a learning algorithm $\mathcal{A}$ and assume there is a corresponding compression scheme (κ, ρ) of size s, such that when $\mathcal{A}$ produces a hypothesis $h_S\colon \mathcal{X} \to \mathcal{Y}$ from a training set S, then the corresponding compression scheme satisfies $\rho(\kappa(S)) = h_S$. In this case, we can prove a bound on the generalization of $h_{\mathbf{S}}$ for a training set $\mathbf{S} \sim \mathcal{D}^n$. In a nutshell, we observe that there are only $M = \sum_{i \leq s} \binom{n}{i}$ possible choices for $\kappa(\mathbf{S})$. Since $\rho(\mathbf{S}')$ for a fixed subset $\mathbf{S}' \subseteq \mathbf{S}$ is determined from the samples in $\mathbf{S}'$ alone, and the remaining $n - |\mathbf{S}'|$ samples are i.i.d. from $\mathcal{D}$, a union bound over the M choices for $\mathbf{S}'$ shows that with probability at least $1 - \delta$, there is no $\mathbf{S}'$ with $\rho(\mathbf{S}')(\mathbf{x}) = \mathbf{y}$ for all $(\mathbf{x}, \mathbf{y}) \in \mathbf{S}$ and yet $R_{\mathcal{D}}(\rho(\mathbf{S}'))$ is larger than $O(\ln(M)/n + \ln(1/\delta)/n) = O((s \ln(n/s) + \ln(1/\delta))/n)$. Thus, in particular, $R_{\mathcal{D}}(h_{\mathbf{S}}) = R_{\mathcal{D}}(\rho(\kappa(\mathbf{S}))) = O((s \ln(n/s) + \ln(1/\delta))/n)$.

Interestingly, the factor $\ln(n/s)$ in the generalization bound can be removed if the compression scheme satisfies an additional property of *stability* introduced by Bousquet et al. (2020). A compression scheme is *stable* if for any training set S and subset S' with $\kappa(S) \subseteq S' \subseteq S$, it holds that $\rho(\kappa(S)) = \rho(\kappa(S'))$. In words, if we remove training samples not part of the compression $\kappa(S)$ from S, then the resulting training set S' is still compressed to the same. Bousquet et al. (2020) proved the first tight generalization bounds for Support Vector Machines by constructing a suitable stable sample compression scheme.

Contribution II: Randomized Compression Schemes. Our work introduces the notion of a randomized compression scheme and use it to prove generalization of Algorithm 1. Such a randomized compression scheme $(\mathcal{D}_\kappa, \rho)$ consists of a distribution $\mathcal{D}_\kappa$ over encoding maps, and a reconstruction function ρ that is not randomized, but simply defined as for regular compression schemes.

As a further extension to the standard compression framework, we give κ an upper bound n of the cardinality of the training sample considered. Furthermore, we allow a bit more freedom in the encoding by not requiring $\kappa(S)$ to be a subsequence of S. More precisely,

- The distribution $\mathcal{D}_\kappa$ is over (deterministic) encoding functions κ that map any sequence $S \in (\mathcal{X} \times \mathcal{Y})^*$ and integer $n \geq |S|$, to a sequence $\kappa(S, n)$ such that every element of $\kappa(S, n)$ appears in S.

We dedicate the symbol "$\sqsubseteq$" to represent that every element of a sequence appears in another sequence. Formally, given sequences $S = (s_1, \ldots, s_m)$ and $T = (t_1, \ldots, t_n)$, we write $S \sqsubseteq T$ if and only if $\{s_i \mid i \in [m]\} \subseteq \{t_j \mid j \in [n]\}$.

Note that the definition above allows the samples in $\kappa(S, n)$ to appear in a different order than in S and to appear a different number of times.

A randomized compression scheme has failure probability at most δ if for all $S \in (\mathcal{X} \times \mathcal{Y})^*$ and $n \geq |S|$ it holds that

$$\Pr_{\boldsymbol{\kappa} \sim \mathcal{D}_\kappa} [\exists (x, y) \in S : \rho(\boldsymbol{\kappa}(S, n))(x) \neq y] \leq \delta.$$

A randomized compression scheme $(\mathcal{D}_\kappa, \rho)$ is *stable* if and only if given i.i.d. $\boldsymbol{\kappa}, \boldsymbol{\kappa}' \sim \mathcal{D}_\kappa$, for any $S \in (\mathcal{X} \times \mathcal{Y})^*$ and $n \in \mathbb{N}$ with $n \geq |S|$, and any subsequence S' of S in the support of $\boldsymbol{\kappa}(S, n)$,

the distribution of $\boldsymbol{\kappa}'(S', n)$ is the same as the distribution of $\boldsymbol{\kappa}(S, n)$ conditioned on $\boldsymbol{\kappa}(S, n) \sqsubseteq S'$. That is, for all $T \in (\mathcal{X} \times \mathcal{Y})^*$, we have that

$$\Pr[\boldsymbol{\kappa}'(S', n) = T] = \Pr[\boldsymbol{\kappa}(S, n) = T \mid \boldsymbol{\kappa}(S, n) \sqsubseteq S'].$$

Given $n \in \mathbb{N}$, the *size* s_n of a randomized compression scheme is the supremum over (S, j) in $\cup_{i=1}^{n}((\mathcal{X} \times \mathcal{Y})^i \times \{i, \ldots, n\})$, and k in the support of $\mathcal{D}_\kappa$, of the number of distinct (x, y) in $\kappa(S, j)$.

Our main technical result for proving generalization via randomized compression is the following theorem:

Theorem 2 *There exists universal constant $C > 0$ for which the following holds. Let $\mathcal{D}$ be an unknown distribution over $\mathcal{X} \times \mathcal{Y}$ and let $\mathbf{S} \sim \mathcal{D}^n$. Let $(\mathcal{D}_\kappa, \rho)$ be a stable randomized compression scheme with failure probability at most δ and size $s = s_n$. Then for every $\beta > 2\delta$, it holds with probability at least $1 - \beta$ over $\mathbf{S}$ and $\boldsymbol{\kappa} \sim \mathcal{D}_\kappa$ that*

$$R_{\mathcal{D}}(\rho(\boldsymbol{\kappa}(\mathbf{S}, n))) \leq C \cdot \frac{s + \ln(1/\beta)}{n},$$

where $R_{\mathcal{D}}(h) = \Pr_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}[h(\mathbf{x}) \neq \mathbf{y}]$.

Similarly to the stable compression schemes of Bousquet et al. (2020), the generalization bound in Theorem 2 depends linearly on s and not as $s \ln(n/s)$ like the bounds of Littlestone and Warmuth (1986) without stability.

In light of Theorem 2, we prove generalization of our new Boosting algorithm, Algorithm 1, by showing that there is a corresponding randomized compression scheme of size $s = s_n = O((d + \ln(1/\gamma)) \ln(n/\delta)/\gamma^4)$ and invoking Theorem 2.

1.2. Main Ideas in Algorithm 1

Having presented our randomized compression framework, let us now discuss the main ideas and obstacles overcome by Algorithm 1 and how they relate to randomized compression. We also argue why the classic compression frameworks are insufficient for our purpose, thus further motivating our randomized framework.

In striving to improve the sample complexity of voting classifiers, a natural approach would be to apply the classic stable compression framework of Bousquet et al. (2020), as it is known to improve sample complexity by a logarithmic factor. However, combining classic sample compression with Boosting appears tricky. To see this, notice that Boosting algorithms invoke a weak learner $\mathcal{W}$ with a distribution D over the full training set S. The weak learner then returns a hypothesis h_D, depending on D, that is used in a final classifier f. For the purpose of invoking a compression framework to argue generalization of f, we would like to argue that a small subset $\kappa(S) \subseteq S$ may be used to reconstruct f. However, we have no control over the weak learner $\mathcal{W}$ and it is completely unclear that we would be able to recover each h_D used in f without including all of S in $\kappa(S)$.

For the reader familiar with AdaBoost, Algorithm 1 is seen to resemble it quite closely. However, for standard AdaBoost, the weak learner $\mathcal{W}$ would be invoked directly on the distributions $\mathbf{D}_k$ in Algorithm 1. In order to give an efficient compression, we instead draw samples $\mathbf{S}_k \sim \mathbf{D}_k^m$ and invoke $\mathcal{W}$ on just the samples. This way, we can intuitively reconstruct the hypotheses $\mathbf{h}_k$ from just the samples $\mathbf{S}_1, \ldots, \mathbf{S}_K$ and this is precisely what we do in our proof of Theorem 1, i.e. we let our encoding be the samples in $\mathbf{S}_1, \ldots, \mathbf{S}_K$.

Still, we need the final classifier produced by Algorithm 1 to be correct on the training data (the compression scheme must have small failure probability). This puts a constraint on the number of samples m and iterations K. Here we use an observation from previous work (Karbasi and Larsen, 2024) on parallel Boosting, showing that the set $\mathbf{S}_k$ forms a $(\gamma/2)$-approximation for the distribution $\mathbf{D}_k$ with good probability (see the correctness proof for details). At a high level, this implies that the hypothesis $\mathbf{h}_k$ returned by the weak learner has error at most $1/2 - \gamma/2$ under $\mathbf{D}_k$. A mostly standard analysis of AdaBoost then shows that after K iterations, the resulting voting classifier $\mathbf{f}$ is correct on all the training data (and thus the compression scheme has small failure probability).

A natural question is whether we really need the randomness from our new framework, or the classic stable compression framework by Bousquet et al. (2020) would suffice. To use their framework, we would need to *deterministically* pick the sets $\mathbf{S}_k$. While it is known that a random $\mathbf{S}_k \sim \mathbf{D}^m$ forms a $\gamma/2$-approximation with constant probability when $m = \Omega(d/\gamma^2)$, it is not clear how to compute such a set deterministically in time less than the number of distinct hypotheses from which the weak learner might choose, which may be as large as $\binom{n}{d}$ when constrained to S.

In light of the above, our new randomized compression framework provides means to analyzing learning algorithms that use random sampling to quickly find sub-samples $\mathbf{S}' \subset S$ with desirable properties that are hard to guarantee deterministically.

Finally, we overview the stability of Algorithm 1 (formal details appear later). That is, we need to argue that for any subsequence $S' \subseteq S$ of the training data, if we condition on $\mathbf{S}_1, \ldots, \mathbf{S}_K \sqsubseteq S'$, then the distribution of $\mathbf{S}_1, \ldots, \mathbf{S}_K$ is the same as the distribution of $\mathbf{S}'_1, \ldots, \mathbf{S}'_K$ resulting from instead running Algorithm 1 on the input S'. We argue this by induction roughly as follows: Assume we have already shown it for the prefix $\mathbf{S}_1, \ldots, \mathbf{S}_k$ and $\mathbf{S}'_1, \ldots, \mathbf{S}'_k$. Then the distribution of the hypotheses $\mathbf{h}_1, \ldots, \mathbf{h}_k$ and $\mathbf{h}'_1, \ldots, \mathbf{h}'_k$ in the two executions would be identical. Now for any $h_1, \ldots, h_k$ in the support of this distribution, the weights in $\mathbf{D}_{k+1}$ and $\mathbf{D}'_{k+1}$ computed by Algorithm 1 are completely determined as $\mathbf{D}_{k+1}(j) = \exp(-y_j \sum_{\ell=1}^{k} \alpha h_\ell(x_j))/Z$ and $\mathbf{D}'_{k+1}(j) = \exp(-y_j \sum_{\ell=1}^{k} \alpha h_\ell(x_j))/Z'$ where Z and Z' are normalization factors making $\mathbf{D}_{k+1}$ and $\mathbf{D}'_{k+1}$ probability distributions. The crucial point is that the "weight" of each point $x_j \in S'$ is the same in $\mathbf{D}_{k+1}$ and $\mathbf{D}'_{k+1}$ up to the normalization terms Z and Z'. When we further condition on $\mathbf{S}_{k+1} \subseteq S'$, this effectively rescales $\mathbf{D}_{k+1}$ by setting all weights outside S' to 0 and changing the normalization factor to Z', making the distribution the same as for $\mathbf{S}'_{k+1}$.

1.3. Other Related Work

Let us finally describe other relevant previous works, in particular results showing barriers for further improving the sample complexity of voting classifiers.

First, one natural approach to training a voting classifier $f(x) = \text{sign}(\sum_t \alpha_t h_t(x))$ with a sample complexity matching the best previously known for voting classifiers (Eq. (1)) is to ensure that f has all *margins* on the training data $\Omega(\gamma)$. The margin of f on a sample (x, y) is defined as

$$\text{margin}_f(x, y) \coloneqq y \cdot \frac{\sum_t \alpha_t h_t(x)}{\sum_t |\alpha_t|}.$$

Margins were originally introduced to explain the excellent practical performance of AdaBoost and its variants (Bartlett et al., 1998). Several *uniform convergence* based generalization bounds have been shown for large margin voting classifiers (Bartlett et al., 1998; Breiman, 1999), with the state-of-the-art being the kth margin bound by Gao and Zhou (2013). Simplified to all margins being

at least γ, they showed that with probability at least $1-\delta$ over a set of n training samples from a distribution $\mathcal{D}$, it simultaneously holds that *all* voting classifiers f with all margins on the training data at least γ satisfy that

$$R_{\mathcal{D}}(f) = O\left(\frac{d\ln(n/d)\ln n}{\gamma^2 n} + \frac{\ln(1/\delta)}{n}\right). \tag{3}$$

Here d denotes the VC-dimension of the hypothesis set $\mathcal{H}$ to which all h_t in the voting classifiers f belong. AdaBoost$^*_\nu$ (Rätsch et al., 2005) is a Boosting algorithm that outputs a voting classifier guaranteed to have all margins $\Omega(\gamma)$. Using Eq. (3) yields the previously best sample complexity of voting classifiers stated in Eq. (1) for the AdaBoost$^*_\nu$ algorithm.[1]

It follows that if the uniform convergence bound for large margin voting classifiers could be strengthened to $O(d/(\gamma^2 n) + \ln(1/\delta)/n)$, then AdaBoost$^*_\nu$ would be an optimal weak-to-strong learner. Unfortunately, lower bounds against uniform convergence (Grønlund et al., 2019, 2020) show example distributions and hypothesis sets such that with constant probability over n samples, *there exists* a voting classifier f with all margins at least γ and yet

$$R_{\mathcal{D}}(f) = \Omega\left(\frac{d\ln(\gamma^2 n/d)}{\gamma^2 n}\right). \tag{4}$$

Abandoning the hope of proving that a voting classifier is optimal via uniform convergence, a natural goal would be to show that a concrete Boosting algorithm, like AdaBoost or AdaBoost$^*_\nu$ is optimal, i.e. to exploit concrete properties of the Boosting algorithm to argue for better generalization than that in Eq. (4). However, recent work (Høgsgaard et al., 2023) shows that all previous Boosting algorithms that produce voting classifiers, satisfy that with constant probability over n samples, the produced voting classifier has a sample complexity of at least that in Eq. (4). At a high level, the work of Høgsgaard et al. (2023) shows that any Boosting algorithm that always invokes the weak learner $\mathcal{W}$ with a distribution $\mathcal{D}$ having support on the full training data set has a generalization error of at least Eq. (4). The only known Boosting algorithms avoiding this pitfall is the optimal, but non-voting classifier, by Larsen and Ritzert (2022), and our new Algorithm 1.

In summary, several barriers need to be overcome to avoid at least one logarithmic factor overhead in the sample complexity as a function of n.

1.4. Preliminaries

Throughout the paper, we assume for simplicity that the training sets contain no duplicates. One can see that this assumption does not reduce the generality of our arguments by, e.g., letting $\mathcal{X}' = \mathcal{X} \times [0,1]$ and changing the input distribution $\mathcal{D}$ to $\mathcal{D}'$ over $\mathcal{X}' \times \mathcal{Y}$, where $\mathcal{D}'$ generates a pair $(\mathbf{x}', \mathbf{y})$ by letting $\mathbf{x}' = (\mathbf{x}, \mathbf{r})$ for $(\mathbf{x}, \mathbf{y}) \sim \mathcal{D}$ and $\mathbf{r} \sim \text{Uniform}([0,1])$. The weak learner then simply ignores $\mathbf{r}$. Finally, as the reader may have noticed, we reserve boldface letters for random variables (e.g., $x \in \mathbb{R}$ vs. $\mathbf{x} \sim \mathcal{N}(0,1)$).

2. Generalization via Randomized Compression

In this section, we prove Theorem 2 which establishes generalization via randomized compression schemes. So, let $\mathbf{S} \sim \mathcal{D}^n$ be a training set of size n and let $s = s_n$.

1. In fact, to prove Theorem 1 we too argue that Algorithm 1 has large margins, leading to the bound being expressed as a minimum by leveraging Eq. (1).

Proof [Proof of Theorem 2] Partition $\mathbf{S}$ into $2s$ buckets of $n/2s$ samples each and denote these buckets by $\mathbf{S}_1, \ldots, \mathbf{S}_{2s}$. For every subset $I \in \binom{[2s]}{s}$ of s indices of buckets, let $\mathbf{S}_I$ denote the concatenation of the samples in buckets $\mathbf{S}_i$ with $i \in I$. Here the notation $\binom{[2s]}{s}$ refers to all subsets of $[2s]$ of cardinality s. Finally, define $\bar{\mathbf{S}}_I$ as the concatenation of the buckets $\mathbf{S}_i$ with $i \notin I$.

Now consider a random $\boldsymbol{\kappa} \sim \mathcal{D}_\kappa$. For each $I \in \binom{[2s]}{s}$, let $E_{I,\boldsymbol{\kappa}}$ denote the event that $\boldsymbol{\kappa}(\mathbf{S}, n) \sqsubseteq \mathbf{S}_I$, which we denote simply as E_I when $\boldsymbol{\kappa}$ is clear from the context. Notice that $\Pr[\cup_I E_I] = 1$ since the size of the compression scheme is s.

Next, for each I and parameter $\alpha > 0$ define $p_{I,\alpha}$ to be the probability

$$\Pr_{\substack{\boldsymbol{\kappa}\sim\mathcal{D}_\kappa,\\ \mathbf{S}_I,\bar{\mathbf{S}}_I\sim\mathcal{D}^{n/2}}} \left[\forall (x,y) \in \bar{\mathbf{S}}_I, \rho(\boldsymbol{\kappa}(\mathbf{S}_I, n))(x) = y \ \wedge \ R_{\mathcal{D}}(\rho(\boldsymbol{\kappa}(\mathbf{S}_I, n))) \geq \alpha\right].$$

To bound $p_{I,\alpha}$, fix any S_I and κ in the supports of $\mathbf{S}_I$ and $\boldsymbol{\kappa}$. If $R_{\mathcal{D}}(\rho(\kappa(S_I, n))) < \alpha$, then S_I and κ contribute 0 to $p_{I,\alpha}$. Otherwise, since $\bar{\mathbf{S}}_I$ is independent of $\mathbf{S}_I$, we have that $\Pr_{\bar{\mathbf{S}}_I\sim\mathcal{D}^{n/2}}[\forall (x,y) \in \bar{\mathbf{S}}_I, \rho(\kappa(\mathbf{S}_I, n))(x) = y] \leq (1-\alpha)^{n/2} \leq \exp(-\alpha n/2)$. Thus $p_{I,\alpha} \leq \exp(-\alpha n/2)$.

Moreover, it holds that

$$\begin{aligned}\Pr_{\substack{\boldsymbol{\kappa}\sim\mathcal{D}_\kappa,\\ \mathbf{S}\sim\mathcal{D}^n}} [R_{\mathcal{D}}(\rho(\boldsymbol{\kappa}(\mathbf{S}, n))) \geq \alpha] \leq \Pr_{\boldsymbol{\kappa},\mathbf{S}}[\exists (x,y) \in \mathbf{S} : \rho(\boldsymbol{\kappa}(\mathbf{S}, n))(x) \neq y] \\ + \Pr_{\boldsymbol{\kappa},\mathbf{S}} \left[\forall (x,y) \in \mathbf{S}, \rho(\boldsymbol{\kappa}(\mathbf{S}, n))(x) = y \ \wedge \ R_{\mathcal{D}}(\rho(\boldsymbol{\kappa}(\mathbf{S}, n))) \geq \alpha\right].\end{aligned}$$

By definition, we have $\Pr[\exists (x,y) \in \mathbf{S} : \rho(\boldsymbol{\kappa}(\mathbf{S}, n))(x) \neq y] < \delta$. Also, since $\cup_I E_I$ always occur,

$$\begin{aligned}&\Pr_{\substack{\boldsymbol{\kappa}\sim\mathcal{D}_\kappa,\\ \mathbf{S}\sim\mathcal{D}^n}} \left[\forall (x,y) \in \mathbf{S}, \rho(\boldsymbol{\kappa}(\mathbf{S}, n))(x) = y \ \wedge \ R_{\mathcal{D}}(\rho(\boldsymbol{\kappa}(\mathbf{S}, n))) \geq \alpha\right] \\ &\quad = \Pr_{\boldsymbol{\kappa},\mathbf{S}} \left[\forall (x,y) \in \mathbf{S}, \rho(\boldsymbol{\kappa}(\mathbf{S}, n))(x) = y \ \wedge \ R_{\mathcal{D}}(\rho(\boldsymbol{\kappa}(\mathbf{S}, n))) \geq \alpha \ \wedge \ \cup_I E_I\right] \\ &\quad \leq \sum_I \Pr_{\boldsymbol{\kappa},\mathbf{S}} \left[\forall (x,y) \in \mathbf{S}, \rho(\boldsymbol{\kappa}(\mathbf{S}, n))(x) = y \ \wedge \ R_{\mathcal{D}}(\rho(\boldsymbol{\kappa}(\mathbf{S}, n))) \geq \alpha \ \wedge \ E_I\right] \\ &\quad = \sum_I \Pr_{\boldsymbol{\kappa},\mathbf{S}} \left[\forall (x,y) \in \mathbf{S}, \rho(\boldsymbol{\kappa}(\mathbf{S}, n))(x) = y \ \wedge \ R_{\mathcal{D}}(\rho(\boldsymbol{\kappa}(\mathbf{S}, n))) \geq \alpha \mid E_I\right] \cdot \Pr_{\boldsymbol{\kappa},\mathbf{S}}[E_I].\end{aligned}$$

Now observe that since $(\mathcal{D}_\kappa, \rho)$ is a stable randomized compression scheme, the distribution of $\rho(\boldsymbol{\kappa}(\mathbf{S}, n))$ conditioned on E_I is the same as $\rho(\boldsymbol{\kappa}'(\mathbf{S}_I, n))$ for a fresh $\boldsymbol{\kappa}' \sim \mathcal{D}_\kappa$. Thus,

$$\begin{aligned}&\sum_I \Pr_{\substack{\boldsymbol{\kappa}\sim\mathcal{D}_\kappa,\\ \mathbf{S}\sim\mathcal{D}^n}} \left[\forall (x,y) \in \mathbf{S}, \rho(\boldsymbol{\kappa}(\mathbf{S}, n))(x) = y \ \wedge \ R_{\mathcal{D}}(\rho(\boldsymbol{\kappa}(\mathbf{S}, n))) \geq \alpha \mid E_I\right] \cdot \Pr_{\boldsymbol{\kappa},\mathbf{S}}[E_I] \\ &\quad = \sum_I \Pr_{\substack{\boldsymbol{\kappa}\sim\mathcal{D}_\kappa,\\ \boldsymbol{\kappa}'\sim\mathcal{D}_\kappa,\\ \mathbf{S}\sim\mathcal{D}^n}} \left[\forall (x,y) \in \mathbf{S}, \rho(\boldsymbol{\kappa}'(\mathbf{S}_I, n))(x) = y \ \wedge \ R_{\mathcal{D}}(\rho(\boldsymbol{\kappa}'(\mathbf{S}_I, n))) \geq \alpha \mid E_{I,\boldsymbol{\kappa}}\right] \cdot \Pr_{\boldsymbol{\kappa},\mathbf{S}}[E_{I,\boldsymbol{\kappa}}] \\ &\quad \leq \sum_I \Pr_{\substack{\boldsymbol{\kappa}'\sim\mathcal{D}_\kappa,\\ \mathbf{S}\sim\mathcal{D}^n}} \left[\forall (x,y) \in \mathbf{S}, \rho(\boldsymbol{\kappa}'(\mathbf{S}_I, n))(x) = y \ \wedge \ R_{\mathcal{D}}(\rho(\boldsymbol{\kappa}'(\mathbf{S}_I, n))) \geq \alpha\right] \\ &\quad \leq \sum_I \Pr_{\substack{\boldsymbol{\kappa}\sim\mathcal{D}_\kappa,\\ \mathbf{S}_I\sim\mathcal{D}^{n/2},\\ \bar{\mathbf{S}}_I\sim\mathcal{D}^{n/2},}} \left[\forall (x,y) \in \bar{\mathbf{S}}_I, \rho(\boldsymbol{\kappa}(\mathbf{S}_I, n))(x) = y \ \wedge \ R_{\mathcal{D}}(\rho(\boldsymbol{\kappa}(\mathbf{S}_I, n))) \geq \alpha\right] \\ &\quad \leq \binom{2s}{s} \exp(-\alpha n/2).\end{aligned}$$

Overall, we conclude that

$$\Pr_{\substack{\boldsymbol{\kappa}\sim\mathcal{D}_\kappa,\\ \mathbf{S}\sim\mathcal{D}^n}}[R_{\mathcal{D}}(\rho(\boldsymbol{\kappa}(\mathbf{S},n))) \geq \alpha] \leq \delta + \binom{2s}{s}\exp(-\alpha n/2).$$

Finally, we obtain the thesis by considering $\beta \geq 2\delta$ and choosing $\alpha = 2(s\ln(4)+\ln(2/\beta))/n$ so that $\binom{2s}{s}\cdot\exp(-\alpha n/2) \leq \beta/2$. ■

3. Efficient Boosting via Randomized Compression

In this section, we present our proof that Algorithm 1 achieves the sample complexity stated in Theorem 1. Recall that we are given access to a γ-weak learner $\mathcal{W}$. For any data set $S \in (\mathcal{X}\times\{-1,1\})^*$ and distribution $\mathcal{D}$ over S, we can query the weak learner with S and $\mathcal{D}$ and it will return a hypothesis $h\colon \mathcal{X}\to\{-1,1\}$ such that $R_{\mathcal{D}}(h) \leq 1/2-\gamma$. We assume that the hypotheses returned by the weak learner belong to a hypothesis set $\mathcal{H}$ of VC-dimension d.

The parameter N in Algorithm 1 is an upper bound on $|S| = n$. It is merely used for sake of analysis when invoking the stable compression framework. It ensures that K remains the same if the algorithm is executed on a subset S' of the training set with the same value of N. When using the algorithm, one should simply set N to n.

At a high level, the algorithm runs AdaBoost with a few twists. We maintain weighted distributions $\mathbf{D}_k$ over the training data. In each step, the weak learner is invoked to obtain a hypothesis $\mathbf{h}_k$ with a small error under distribution $\mathbf{D}_k$. However, unlike in AdaBoost, we do not invoke the weak learner on the full training data. Instead, we obtain $\mathbf{h}_k$ by sampling some $m = O((d+\ln(1/\gamma))\gamma^{-2})$ data points, denoted $\mathbf{S}_k$, from $\mathbf{D}_k$ and train on $\mathbf{S}_k$ with a uniform weighing. Furthermore, where AdaBoost would normally update all weights by e^{α} or $e^{-\alpha}$ for $\alpha = \alpha_k = (1/2)\ln((1-R_{\mathbf{D}_k}(\mathbf{h}_k))/R_{\mathbf{D}_k}(\mathbf{h}_k))$, we simply fix α as if $R_{\mathbf{D}_k}(\mathbf{h}_k)$ was $1/2-\gamma/2$.

3.1. Corresponding Randomized Compression Scheme

We now argue that Algorithm 1 naturally corresponds to a randomized compression scheme. Let $S = \big((x_1,y_1),\ldots,(x_n,y_n)\big)$ be the training sequence and $N \geq n$. Consider an execution of the randomized Algorithm 1 and let $\mathbf{h}_1,\ldots,\mathbf{h}_K$ be the hypotheses obtained. From such an execution, we define an encoding map $\boldsymbol{\kappa}$ that maps (S,N) to the sequence $\mathbf{S}_1\circ\cdots\circ\mathbf{S}_K$, where $\circ$ denotes concatenation and $\mathbf{S}_i$ is the sample associated with $\mathbf{h}_i$ (see Line 6). The randomized algorithm thus gives a distribution $\mathcal{D}_\kappa$ over such encoding maps.

Our reconstruction function ρ on a sequence of $K\cdot m$ samples partitions the samples into K consecutive groups $S_1,\ldots,S_K$ of m samples. It then invokes the weak learner $\mathcal{W}$ on each S_i with the uniform distribution to obtain h_i and finally produces the function mapping any $x\in\mathcal{X}$ to $\operatorname{sign}((1/K)\sum_{k=1}^{K} h_k(x))$.

Notice that $\rho(\boldsymbol{\kappa}(S,N))(x) = \operatorname{sign}(\mathbf{f}(x))$, i.e. the reconstruction function makes the same predictions as the returned voting classifier. Hence if we can show that the obtained randomized compression scheme has a small failure probability and is stable, then we may use Theorem 2 to bound the generalization error of Algorithm 1. In particular, our compression scheme has size $O(Km)$. Combining this bound on the size with Theorem 2 proves Theorem 1.

In the following, we first argue that the obtained compression scheme has failure probability at most δ (Lemma 3). We then argue that it is indeed stable (Lemma 5).

3.2. Small Failure Probability

We show that for any training set S, with good probability over the execution of Algorithm 1 with $N \geq |S| = n$, the returned voting classifier $\mathbf{f}(x) = (1/K)\sum_{i=1}^{K} \mathbf{h}_i(x)$ has large margins on all the training data S. Thus, we can apply Eq. 3 to it. Moreover, this also implies that $\operatorname{sign}(\mathbf{f})$ has zero empirical error, bounding the failure probability of the algorithm. Concretely, we show:

Lemma 3 *For any training set $S = ((x_1, y_1), \ldots, (x_n, y_n))$, it holds with probability at least $1-\delta$ over the execution of Algorithm 1 with $N \geq n$ that the voting classifier $\mathbf{f}(x) = (1/K)\sum_{i=1}^{K} \mathbf{h}_i(x)$ satisfies, for all $i \in [n]$, that $y_i\mathbf{f}(x_i) \geq \gamma/128$, and, in particular, that $\operatorname{sign}(\mathbf{f}(x_i)) = y_i$.*

The proof of Lemma 3 makes use of the notion of an ε-approximation. For a concept $c\colon \mathcal{X} \to \{-1, 1\}$, a hypothesis set $\mathcal{H}$ and a distribution $\mathcal{D}$ over $\mathcal{X}$, a set of samples S is an ε-approximation for $(c, \mathcal{D}, \mathcal{H})$ if for all $h \in \mathcal{H}$, it holds that

$$\left| \Pr_{x\sim\mathcal{D}}[h(x) \neq c(x)] - \frac{|\{x \in S : h(x) \neq c(x)\}|}{|S|} \right| \leq \varepsilon.$$

The following result ensures that a large enough set of samples $\mathbf{S} \sim \mathcal{D}^n$ is an ε-approximation with good probability.

Theorem 4 (Li et al. 2001; Talagrand 1994; Vapnik and Chervonenkis 1971) *There exists universal constant $b > 0$, such that for any $0 < \varepsilon, \delta < 1$, any concept $c\colon \mathcal{X} \to \{-1, 1\}$, any $\mathcal{H} \subseteq \mathcal{X} \to \{-1, 1\}$ of VC-dimension d and any distribution $\mathcal{D}$ over $\mathcal{X}$, it holds with probability at least $1-\delta$ over a set $\mathbf{S} \sim \mathcal{D}^n$ that $\mathbf{S}$ is an ε-approximation for $(c, \mathcal{D}, \mathcal{H})$ provided that $n \geq b((d + \ln(1/\delta))\varepsilon^{-2})$.*

We now present our formal argument.

Proof [of Lemma 3] Fix any set S of n samples $(x_1, y_1), \ldots, (x_n, y_n)$ and let $c\colon (\mathcal{X} \cap S) \to \{-1, 1\}$ denote the concept with $c(x_i) = y_i$ for each $i = 1, \ldots, n$.

Define an indicator random variable $\mathbf{X}_k$ for each step $k = 1, \ldots, K$ taking the value 1 if $\mathbf{S}_k$ fails to be a $\gamma/2$-approximation for $(c, \mathbf{D}_k, \mathcal{H})$. Note that for any outcome $S_1, \ldots, S_{k-1}$ of the random samples $\mathbf{S}_1, \ldots, \mathbf{S}_{k-1}$, we get from Theorem 4 and our choice of $m = a((d + \ln(1/\gamma))\gamma^{-2})$ that $\Pr[\mathbf{X}_k = 1 \mid \forall i < k : \mathbf{S}_i = S_i] \leq \gamma^2/32$ for a large enough constant $a > 0$. It follows from a Chernoff bound that $\Pr[\sum_i \mathbf{X}_i > \gamma^2 K/16] \leq \exp(-\gamma^2 K/32) = \delta/(eN) < \delta/2$. Let us now assume that at most $\gamma^2 K/16$ of the samples $\mathbf{S}_i$ fail to be a $\gamma/2$-approximation. We claim that $\mathbf{f}(x) = (1/K)\sum_{k=1}^{K} \mathbf{h}_k(x)$ satisfies $y_i\mathbf{f}(x_i) \geq \gamma/128$ in this case.

To see this, consider the exponential loss

$$\sum_{i=1}^{n} \exp\left(-\alpha y_i \sum_{k=1}^{K} \mathbf{h}_k(x_i)\right).$$

We compare this to the final weights $\mathbf{D}_{K+1}$. Since $\mathbf{D}_{K+1}$ is a probability distribution, we have

$$\begin{aligned}
1 &= \sum_{i=1}^{n} \mathbf{D}_{K+1}(i) \\
&= \sum_{i=1}^{n} \frac{\mathbf{D}_K(i) \exp(-\alpha y_i \mathbf{h}_K(x_i))}{\mathbf{Z}_k} \\
&= \frac{1}{n} \sum_{i=1}^{n} \frac{\exp(-\alpha y_i \sum_{k=1}^{K} \mathbf{h}_k(x_i))}{\prod_{k=1}^{K} \mathbf{Z}_k}.
\end{aligned}$$

From this, we observe that

$$\sum_{i=1}^{n} \exp\left(-\alpha y_i \sum_{k=1}^{K} \mathbf{h}_k(x_i)\right) = n \prod_{k=1}^{K} \mathbf{Z}_k.$$

To bound the $\mathbf{Z}_k$, we analyze two cases. First, if $\mathbf{X}_k = 0$, then we know that $\mathbf{S}_k$ is a $\gamma/2$-approximation for $\mathbf{D}_k$. Furthermore, since $\mathcal{W}$ is a γ-weak learner, we have that $R_{\mathbf{S}_k}(\mathbf{h}_k) \leq 1/2 - \gamma$ where $R_{\mathbf{S}_k}(\mathbf{h}_k)$ denotes the fraction of mispredictions among samples in $\mathbf{S}_k$. By the definition of a $\gamma/2$-approximation, this further implies $R_{\mathbf{D}_k}(\mathbf{h}_k) \leq 1/2 - \gamma/2$. If $\mathbf{X}_k = 1$, then we simple bound $R_{\mathbf{D}_k}(\mathbf{h}_k) \leq 1$.

We now observe that

$$\begin{aligned}
\mathbf{Z}_k &= \sum_{i=1}^{m} \mathbf{D}_k(i) \exp(-\alpha y_i \mathbf{h}_k(x_i)) \\
&= \sum_{i:\mathbf{h}_k(x_i) \neq y_i} \mathbf{D}_k(i) e^{\alpha} + \sum_{i:\mathbf{h}_k(x_i) = y_i} \mathbf{D}_k(i) e^{-\alpha} \\
&= R_{\mathbf{D}_k}(\mathbf{h}_k) e^{\alpha} + (1 - R_{\mathbf{D}_k}(\mathbf{h}_k)) e^{-\alpha}.
\end{aligned}$$

For $\mathbf{X}_k = 0$, this is upper bounded by

$$\begin{aligned}
\mathbf{Z}_k &\leq (1/2 - \gamma/2) e^{\alpha} + (1/2 + \gamma/2) e^{-\alpha} \\
&= 2\sqrt{(1/2 - \gamma/2)(1/2 + \gamma/2)} \\
&= \sqrt{1 - \gamma^2}.
\end{aligned}$$

For $\mathbf{X}_k = 1$, it is upper bounded by

$$\begin{aligned}
\mathbf{Z}_k &\leq e^{\alpha} \\
&= \sqrt{(1/2 + \gamma/2)/(1/2 - \gamma/2)} \\
&\leq \sqrt{1 + \frac{\gamma}{1/2 - \gamma/2}} \\
&\leq \sqrt{1 + 4\gamma}.
\end{aligned} \tag{5}$$

Using that $\sum_{k=1}^{K} \mathbf{X}_k \leq \gamma^2 K/16$, we thus conclude

$$\begin{aligned}\prod_{k=1}^{K} \mathbf{Z}_k &\leq (1-\gamma^2)^{(K-\gamma^2 K/16)/2}(1+4\gamma)^{\gamma^2 K/32} \\ &\leq \exp\left(\gamma^3 K/8 - \gamma^2(K-\gamma^2 K/16)/2\right) \\ &\leq \exp(-\gamma^2 K/4) \\ &\leq (\delta/N)^2.\end{aligned}$$

We therefore have

$$\sum_{i=1}^{n} \exp\left(-\alpha y_i \sum_{k=1}^{K} \mathbf{h}_k(x_i)\right) \leq \delta/N,$$

so, by non-negativity of the exponential function, $\exp(-\alpha y_i \sum_{k=1}^{K} \mathbf{h}_k(x_i)) \leq \delta/N$ for all $i \in [n]$. Raising both sides of the inequality to the power $1/(K\alpha)$ gives $\exp(-y_i \mathbf{f}(x_i)) \leq (\delta/N)^{1/K\alpha}$, so $y_i \mathbf{f}(x_i) \geq \ln(N/\delta)/(K\alpha)$. From Eq. (5), we have that $e^{\alpha} \leq \sqrt{1+4\gamma}$, hence $\alpha \leq (1/2)\ln(1+4\gamma) \leq (1/2)\ln(e^{4\gamma}) = 2\gamma$. Thus, we conclude that $y_i \mathbf{f}(x_i) \geq \ln(N/\delta)/(K2\gamma) \geq \gamma/128$. ■

3.3. Stability

In the following, we show the stability of the compression scheme corresponding to Algorithm 1.

Fix a γ-weak learner $\mathcal{W}$, a failure probability δ, and an upper bound N on the size of the training set. Given $S \in \cup_{i=1}^{N} (\mathcal{X} \times \mathcal{Y})^i$, let $\textsc{Exec}(S, N) = \mathbf{S}_1, \ldots, \mathbf{S}_K$ denote the sequence of samples associated with the execution of Algorithm 1 on input $S, \mathcal{W}, \delta, N$. In this way, the sequence $\mathbf{S}_i$ is the sample drawn at Line 6 on the ith iteration of the **for** loop starting at Line 5. The randomized compression scheme $\boldsymbol{\kappa}$ underlying Algorithm 1, as discussed in Section 3.1, can then be described by $\boldsymbol{\kappa}(S, N) = \mathbf{S}_1 \circ \cdots \circ \mathbf{S}_K$.

Lemma 5 *The randomized compression scheme $\boldsymbol{\kappa}$ given by $\boldsymbol{\kappa}(S, N) = \textsc{Exec}(S, N)$ is stable.*

Proof Given $n \in [N]$, let $S \in (\mathcal{X} \times \mathcal{Y})^n$, and let S' be a subsequence of S. Let $\textsc{Exec}(S, N) = \mathbf{S}_1, \ldots, \mathbf{S}_K$ and $\textsc{Exec}(S', N) = \mathbf{S}'_1, \ldots, \mathbf{S}'_K$. We will show that for all $k \in [K]$ it holds that conditioning on $\mathbf{S}_i \sqsubseteq S'$ for $i \in [k]$ implies that $\mathbf{S}_1 \circ \cdots \circ \mathbf{S}_k$ follows the same distribution as $\mathbf{S}'_1 \circ \cdots \circ \mathbf{S}'_k$. We argue by induction on k and conclude the thesis by considering $k = K$.

For the base case, we have that $\mathbf{S}_1$ consists of m i.i.d. samples from the uniform distribution over S. Therefore, conditioning on $\mathbf{S}_1 \sqsubseteq S'$ makes the m samples i.i.d. following the uniform distribution over S' and, thus, makes $\mathbf{S}_1$ identically distributed to $\mathbf{S}'_1$ (this uses our assumption that S contains no duplicates).

Now, for the induction step, suppose that for some $k \in [K-1]$ we have that, for all $T \sqsubseteq S$,

$$\Pr\left[\mathbf{S}_1 \circ \cdots \circ \mathbf{S}_k = T \;\middle|\; \mathbf{S}_i \sqsubseteq S' \; \forall i \in [k]\right] = \Pr\left[\mathbf{S}'_1 \circ \cdots \circ \mathbf{S}'_k = T\right].$$

We consider $T \sqsubseteq S'$ since otherwise both sides of the equation are zero. For $i \in [k+1]$, let $\mathbf{D}_i$ and $\mathbf{h}_i$ be the distribution (see Line 6) and hypothesis (see Line 7) corresponding to the ith iteration of the **for** loop starting at Line 5 when executing Algorithm 1 on input $S, \mathcal{W}, \delta, N$. Define $\mathbf{D}'_i$s and $\mathbf{h}'_i$s associated with the execution on $S', \mathcal{W}, \delta, N$ analogously.

For the remainder of the proof, we condition on the event that $\mathbf{S}_i \sqsubseteq S'$ for all $i \in [k]$. The induction hypothesis implies that $\mathbf{S}_1, \dots, \mathbf{S}_k$ and $\mathbf{S}'_1, \dots, \mathbf{S}'_k$ follow the same distribution. Now fix any $T_k = S_1, \dots, S_k$ in the support of this distribution. Note that conditioning on T_k fixes the hypotheses $\mathbf{h}_1, \dots, \mathbf{h}_k$ and $\mathbf{h}'_1, \dots, \mathbf{h}'_k$ to the same fixed $h_1, \dots, h_k$. This further fixes $\mathbf{D}_{k+1}$ to $D_{k+1}(j) = \exp(-\alpha y_j \sum_{\ell=1}^{k} h_\ell(x_j))/Z$ where Z is a normalization factor making D_{k+1} a probability distribution. Similarly for S', it fixes $\mathbf{D}'_{k+1}$ to $D'_{k+1}(j) = \exp(-\alpha y_j \sum_{\ell=1}^{k} h_\ell(x_j))/Z'$ for the $j \in S'$.

The crucial observation is that any x_j occurring in both S' and S have the same weight in D_{k+1} and D'_{k+1} up to the normalization factors Z and Z'. This implies that if we further condition on $\mathbf{S}_{k+1} \sqsubseteq S'$, the samples in $\mathbf{S}_{k+1}$ are i.i.d. from D_{k+1} but where every $j \notin S'$ has $D_{k+1}(j) = 0$ and the resulting distribution is scaled accordingly. This makes the distribution identical to D'_{k+1} (using the assumption that S contains no duplicates), which concludes the proof. ■

4. Conclusion

In this work, we took a first step towards developing voting classifiers with an optimal sample complexity for weak-to-strong learning. Concretely, we improve the dependency on the number of samples n by a logarithmic factor over previous works. To analyze our new algorithm, we further introduce a new framework of randomized compression schemes that we hope may prove useful in future work.

Our work leaves open a number of intriguing directions to pursue. First, can we develop a voting classifier with an optimal sample complexity as in Eq. (2)? Or, as a first and more modest goal, can we develop a voting classifier with only a single logarithmic sub-optimal dependency on n, like our Algorithm 1, but with an optimal dependency on the remaining parameters d, γ, and δ? Another question is whether our analysis of Algorithm 1 is tight, or could it perhaps be improved to yield an even better sample complexity? Also, for previous algorithms such as AdaBoost, the current best analysis gives a sample complexity as in Eq. (1) with two logarithmic factors of sub-optimality. Can the analysis be improved for some of those algorithms? We know that it can never be improved to an optimal sample complexity (in light of (Høgsgaard et al., 2023), see the discussion in Section 1.3), but perhaps one of the logarithmic factors can be removed. The same holds for the uniform convergence bounds for large-margin voting classifiers. Can these be improved by a logarithmic factor?

Acknowledgments

This research is co-funded by the European Union (ERC, TUCLA, 101125203) and Independent Research Fund Denmark (DFF) Sapere Aude Research Leader Grant No. 9064-00068B. 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. Neither the European Union nor the granting authority can be held responsible for them.

Parts of this research was done while Martin Ritzert was supported by DIREC – Digital Research Centre Denmark.

References

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

Olivier Bousquet, Steve Hanneke, Shay Moran, and Nikita Zhivotovskiy. Proper learning, helly number, and an optimal svm bound. In *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*, volume 125 of *Proceedings of Machine Learning Research*, pages 582–609. PMLR, 2020.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

Leo Breiman. Prediction games and arcing algorithms. *Neural computation*, 11(7):1493–1517, 1999.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *KDD*, pages 785–794. ACM, 2016. ISBN 978-1-4503-4232-2.

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, 1997.

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189 – 1232, 2001.

Wei Gao and Zhi-Hua Zhou. On the doubt about margin explanation of boosting. *Artif. Intell.*, 203: 1–18, 2013.

Allan Grønlund, Lior Kamma, Kasper Green Larsen, Alexander Mathiasen, and Jelani Nelson. Margin-based generalization lower bounds for boosted classifiers. *Advances in Neural Information Processing Systems*, 32, 2019.

Allan Grønlund, Lior Kamma, and Kasper Green Larsen. Margins are insufficient for explaining gradient boosting. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 2020.

Mikael Møller Høgsgaard, Kasper Green Larsen, and Martin Ritzert. Adaboost is not an optimal weak to strong learner. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 13118–13140. PMLR, 2023.

Amin Karbasi and Kasper Green Larsen. The impossibility of parallelizing boosting. In *International Conference on Algorithmic Learning Theory, ALT*, 2024. To appear.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *NIPS*, 2017.

Michael Kearns. Learning boolean formulae or finite automata is as hard as factoring. *Technical Report TR-14-88 Harvard University Aikem Computation Laboratory*, 1988.

Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1):67–95, 1994.

Kasper Green Larsen. Bagging is an optimal PAC learner. *Conference on Learning Theory (COLT 2023)*, 195:450–468, 2023.

Kasper Green Larsen and Martin Ritzert. Optimal weak to strong learning. *Advances in Neural Information Processing Systems (NeurIPS 2022)*, 2022.

Y. Li, P.M. Long, and A. Srinivasan. Improved bounds on the sample complexity of learning. *Journal of Computer and System Sciences*, 62:516 – 527, 2001.

N. Littlestone and M Warmuth. Relating data compression and learnability. *Unpublished manuscript*, 1986.

Shay Moran and Amir Yehudayoff. Sample compression schemes for VC classes. *J. ACM*, 63(3): 21:1–21:10, 2016. doi: 10.1145/2890490. URL https://doi.org/10.1145/2890490.

Alexey Natekin and Alois Knoll. Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics*, 7, 2013. ISSN 1662-5218.

Gunnar Rätsch, Manfred K Warmuth, and John Shawe-Taylor. Efficient margin maximizing with boosting. *Journal of Machine Learning Research*, 6(12), 2005.

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

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

M. Talagrand. Sharper Bounds for Gaussian and Empirical Processes. *The Annals of Probability*, 22(1):28 – 76, 1994.

V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.