Online Gradient Boosting

Alina Beygelzimer
Yahoo Labs
New York, NY 10036
beygel@yahoo-inc.com

Elad Hazan
Princeton University
Princeton, NJ 08540
ehazan@cs.princeton.edu

Satyen Kale
Yahoo Labs
New York, NY 10036
satyen@yahoo-inc.com

Haipeng Luo
Princeton University
Princeton, NJ 08540
haipengl@cs.princeton.edu

June 16, 2015

Abstract

We extend the theory of boosting for regression problems to the online learning setting. Generalizing from the batch setting for boosting, the notion of a weak learning algorithm is modeled as an online learning algorithm with linear loss functions that competes with a base class of regression functions, while a strong learning algorithm is an online learning algorithm with convex loss functions that competes with a larger class of regression functions. Our main result is an online gradient boosting algorithm which converts a weak online learning algorithm into a strong one where the larger class of functions is the linear span of the base class. We also give a simpler boosting algorithm that converts a weak online learning algorithm into a strong one where the larger class of functions is the convex hull of the base class, and prove its optimality.

1 Introduction

Boosting algorithms [21] are ensemble methods that convert a learning algorithm for a base class of models with weak predictive power, such as decision trees, into a learning algorithm for a class of models with stronger predictive power, such as a weighted majority vote over base models in the case of classification, or a linear combination of base models in the case of regression.

Boosting methods such as AdaBoost [9] and Gradient Boosting [10] have found tremendous practical application, especially using decision trees as the base class of models. These algorithms were developed in the batch setting, where training is done over a fixed batch of sample data. However, with the recent explosion of huge data sets which do not fit in main memory, training in the batch setting is infeasible, and online learning techniques which train a model in one pass over the data have proven extremely useful.

A natural goal therefore is to extend boosting algorithms to the online learning setting. Indeed, there has already been some work on online boosting for classification problems [20, 11, 17, 12, 4, 5, 2]. Of these, the work by Chen et al. [4] provided the first theoretical study of online boosting for classification, which was later generalized by Beygelzimer et al. [2] to obtain optimal and adaptive online boosting algorithms.

However, extending boosting algorithms for regression to the online setting has been elusive and escaped theoretical guarantees thus far. In this paper, we rigorously formalize the setting of online boosting for regression and then extend the very commonly used gradient boosting methods [10, 19] to the online setting, providing theoretical guarantees on their performance.

The main result of this paper is an online boosting algorithm that competes with any linear combination the base functions, given an online linear learning algorithm over the base class. This algorithm is the online analogue of the batch boosting algorithm of Zhang and Yu [24], and in fact our algorithmic technique, when specialized to the batch boosting setting, provides exponentially better convergence guarantees.

We also give an online boosting algorithm which competes the best convex combination of base functions. This is a simpler algorithm which is analyzed along the lines of the Frank-Wolfe algorithm [8]. While the algorithm has weaker theoretical guarantees, in practical settings this algorithm can still be useful. We also prove that this algorithm obtains the optimal regret bound (up to constant factors) for this setting.

Finally, we conduct some proof-of-concept experiments which show that our online boosting algorithms do obtain performance improvements over different classes of base learners.

1.1 Related Work

While the theory of boosting for classification in the batch setting is well-developed (see [21]), corresponding literature for theory of boosting for regression, particularly including proofs of convergence based on natural assumptions on the base learning algorithm, is comparatively sparse. The foundational theory of boosting for regression can be found in the statistics literature [14, 13], where boosting is understood as a greedy stagewise algorithm for fitting of additive models. The goal is achieve the performance of linear combinations of base models, and to prove convergence to the performance of the best such linear combination.

While the earliest works on boosting for regression such as [10] do not have such convergence proofs, later works such as [19, 6] do have convergence proofs but without a bound on the speed of convergence. Bounds on the speed of convergence have been obtained by Duffy and Helmbold [7] relying on a somewhat strong assumption on the performance of the base learning algorithm. A different approach to boosting for regression was taken by Freund and Schapire [9], who give an algorithm that reduces the regression problem to classification and then applies AdaBoost; the corresponding proof of convergence relies on an assumption of the induced classification problem which may be hard to satisfy in practice. The strongest result is that of Zhang and Yu [24], who prove convergence to the performance of the best linear combination of base functions, along with a bound on the rate of convergence, making essentially no assumptions on the performance of the base learning algorithm. Telgarsky [22] proves similar results for logistic (or similar) loss using a slightly simpler boosting algorithm.

The results in this paper are a generalization of the results of Zhang and Yu [24] to the online setting. However, we emphasize that this generalization is nontrivial and requires different algorithmic ideas and proof techniques. Indeed, we were not able to directly generalize the analysis in [24] by simply adapting the techniques used in recent online boosting work [4, 2], but we made use of the classical Frank-Wolfe algorithm [8]. On the other hand, while an important part of the convergence analysis for the batch setting is to show statistical consistency of the algorithms [24, 1, 22], in the online setting we only need to study the empirical convergence (that is, the regret), which makes our analysis much more concise.

2 Setup

Examples are chosen from a feature space $\mathcal{X}$, and the prediction space is $\mathbb{R}^d$. Let $\|\cdot\|$ denote some norm in $\mathbb{R}^d$. In the setting for online regression, in each round t for $t = 1, 2, \ldots, T$, an adversary selects an example $\mathbf{x}_t \in \mathcal{X}$ and a loss function $\ell_t : \mathbb{R}^d \to \mathbb{R}$, and presents $\mathbf{x}_t$ to the online learner. The online learner outputs a prediction $\mathbf{y}_t \in \mathbb{R}^d$, obtains the loss function ℓ_t, and incurs loss $\ell_t(\mathbf{y}_t)$.

Let $\mathcal{F}$ denote a reference class of regression functions $f : \mathcal{X} \to \mathbb{R}^d$, and let $\mathcal{C}$ denote a class of loss functions $\ell : \mathbb{R}^d \to \mathbb{R}$. Also, let $R : \mathbb{N} \to \mathbb{R}_+$ be a non-decreasing function. We say that the function class $\mathcal{F}$ is *online learnable* for losses in $\mathcal{C}$ with regret R if there is an online learning algorithm $\mathcal{A}$, that for every $T \in \mathbb{N}$ and every sequence $(\mathbf{x}_t, \ell_t) \in \mathcal{X} \times \mathcal{C}$ for $t = 1, 2, \ldots, T$ chosen by the adversary, generates predictions[1] $\mathcal{A}(\mathbf{x}_t) \in \mathbb{R}^d$ such that

$$\sum_{t=1}^{T} \ell_t(\mathcal{A}(\mathbf{x}_t)) \;\leq\; \inf_{f \in \mathcal{F}} \sum_{t=1}^{T} \ell_t(f(\mathbf{x}_t)) + R(T). \tag{1}$$

If the online learning algorithm is randomized, we require the above bound to hold with high probability.

A concrete example of this setting is 1-dimensional regression, i.e. the prediction space is $\mathbb{R}$, where in each round, the adversary chooses a labeled data point $(\mathbf{x}_t, y_t^\star) \in \mathcal{X} \times \mathbb{R}$, and the loss for the prediction $y_t \in \mathbb{R}$ is given by $\ell_t(y_t) = \ell(y_t^\star, y_t)$ where $\ell(\cdot, \cdot)$ is a fixed loss function that is convex in the second argument (such as squared loss, logistic loss, etc). If $\mathcal{X} = \mathbb{R}^n$ for some n, the class $\mathcal{F}$ could be the set of all bounded-norm linear regressors, for example, and the algorithm $\mathcal{A}$ could be online gradient descent [25] or online Newton Step [16].

Informally, an online boosting algorithm is a reduction that, given access to an online learning algorithm $\mathcal{A}$ for a function class $\mathcal{F}$ and loss function class $\mathcal{C}$ with regret R, and a bound N on the total number of calls made in each iteration to copies of $\mathcal{A}$, obtains an online learning algorithm $\mathcal{A}'$ for a richer function class $\mathcal{F}'$, a richer loss function class $\mathcal{C}'$, and (possibly larger) regret R'. The bound N on the total number of calls made to all the copies of $\mathcal{A}$ corresponds to the number of boosting stages in the batch setting, and in the online setting it may be viewed as a resource constraint on the algorithm. The efficacy of the reduction is measured by R' which is a function of R, N, and certain parameters of the comparator class $\mathcal{F}'$ and loss function class $\mathcal{C}'$. We desire online boosting algorithms such that $\frac{1}{T}R'(T) \to 0$ quickly as $N \to \infty$ and $T \to \infty$. We make the notions of richness in the above informal description more precise now.

Comparator function classes. A given function class $\mathcal{F}$ is said to be D-bounded if for all $\mathbf{x} \in \mathcal{X}$ and all $f \in \mathcal{F}$, we have $\|f(\mathbf{x})\| \leq D$. Throughout this paper, we assume that $\mathcal{F}$ is symmetric:[2] i.e. if $f \in \mathcal{F}$, then $-f \in \mathcal{F}$, and it contains the constant zero function, which we denote, with some abuse of notation, by $\mathbf{0}$.

[1] There is a slight abuse of notation here. $\mathcal{A}(\cdot)$ is not a function but rather the output of the online learning algorithm $\mathcal{A}$ computed on the given example using its internal state.

[2] This is without loss of generality; as will be seen momentarily, our base assumption only requires an online learning algorithm $\mathcal{A}$ for $\mathcal{F}$ for linear losses ℓ_t. By running the Hedge algorithm on two copies of $\mathcal{A}$, one of which receives the actual loss functions ℓ_t and the other recieves $-\ell_t$, we get an algorithm which competes with negations of functions in $\mathcal{F}$ and the constant zero function as well. Furthermore, since the loss functions are convex (indeed, linear) this can be made into a deterministic reduction by choosing the convex combination of the outputs of the two copies of $\mathcal{A}$ with mixing weights given by the Hedge algorithm.

Given $\mathcal{F}$, we define two richer function classes $\mathcal{F}'$: the convex hull of $\mathcal{F}$, denoted $\mathrm{CH}(\mathcal{F})$, is the set of convex combinations of a finite number of functions in $\mathcal{F}$, and the span of $\mathcal{F}$, denoted $\mathrm{span}(\mathcal{F})$, is the set of linear combinations of finitely many functions in $\mathcal{F}$. For any $f \in \mathrm{span}(\mathcal{F})$, define $\|f\|_1 := \inf \Big\{ \max\{1, \sum_{g \in S} |w_g|\} : \ f = \sum_{g \in S} w_g g, \ S \subseteq \mathcal{F}, \ |S| < \infty, \ w_g \in \mathbb{R} \Big\}$. Since functions in $\mathrm{span}(\mathcal{F})$ are not bounded, it is not possible to obtain a uniform regret bound for all functions in $\mathrm{span}(\mathcal{F})$: rather, the regret of an online learning algorithm $\mathcal{A}$ for $\mathrm{span}(\mathcal{F})$ is specified in terms of regret bounds for individual comparator functions $f \in \mathrm{span}(F)$, viz.

$$R_f(T) \ := \ \sum_{t=1}^{T} \ell_t(\mathcal{A}(\mathbf{x}_t)) - \sum_{t=1}^{T} \ell_t(f(\mathbf{x}_t)).$$

Loss function classes. The base loss function class we consider is $\mathcal{L}$, the set of all linear functions $\ell : \mathbb{R}^d \to \mathbb{R}$, with Lipschitz constant bounded by 1. A function class $\mathcal{F}$ that is online learnable with the loss function class $\mathcal{L}$ is called *online linear learnable* for short. The richer loss function class we consider is denoted by $\mathcal{C}$ and is a set of convex loss functions $\ell : \mathbb{R}^d \to \mathbb{R}$ satisfying some regularity conditions specified in terms of certain parameters described below.

We define a few parameters of the class $\mathcal{C}$. For any $b > 0$, let $\mathbb{B}^d(b) = \{\mathbf{y} \in \mathbb{R}^d : \ \|\mathbf{y}\| \le b\}$ be the ball of radius b. The class $\mathcal{C}$ is said to have Lipschitz constant L_b on $\mathbb{B}^d(b)$ if for all $\ell \in \mathcal{C}$ and all $\mathbf{y} \in \mathbb{B}^d(b)$ there is an efficiently computable subgradient $\nabla\ell(\mathbf{y})$ with norm at most L_b. Next, $\mathcal{C}$ is said to be β_b-smooth on $\mathbb{B}^d(b)$ if for all $\ell \in \mathcal{C}$ and all $\mathbf{y}, \mathbf{y}' \in \mathbb{B}^d(b)$ we have

$$\ell(\mathbf{y}') \ \le \ \ell(\mathbf{y}) + \nabla\ell(\mathbf{y}) \cdot (\mathbf{y}' - \mathbf{y}) + \frac{\beta_b}{2}\|\mathbf{y} - \mathbf{y}'\|^2.$$

Next, define the projection operator $\Pi_b : \mathbb{R}^d \to \mathbb{B}^d(b)$ as $\Pi_b(\mathbf{y}) := \arg\min_{\mathbf{y}' \in \mathbb{B}^d(b)} \|\mathbf{y} - \mathbf{y}'\|$, and define $\epsilon_b := \sup_{\mathbf{y} \in \mathbb{R}^d, \ \ell \in \mathcal{C}} \frac{\ell(\Pi_b(\mathbf{y})) - \ell(\mathbf{y})}{\|\Pi_b(\mathbf{y}) - \mathbf{y}\|}$.

Main results. The setup is that we are given a D-bounded reference class of functions $\mathcal{F}$ with an online linear learning algorithm $\mathcal{A}$ with regret bound $R(\cdot)$. For normalization, we also assume that the output of $\mathcal{A}$ at any time is bounded in norm by D, i.e. $\|\mathcal{A}(\mathbf{x}_t)\| \le D$ for all t. We further assume that for every $b > 0$, we can compute[3] a Lipschitz constant L_b, a smoothness parameter β_b, and the parameter ϵ_b for the class $\mathcal{C}$ over $\mathbb{B}^d(b)$. Furthermore, the online boosting algorithm may make up to N calls total per iteration to any copies of $\mathcal{A}$ it maintains, for a given a budget parameter N.

Given this setup, our main result is a boosting algorithm, Algorithm 1, competing with $\mathrm{span}(\mathcal{F})$ that has the following regret bound, proved in Section 3.1.

Theorem 1. *Let $\eta \in [\frac{1}{N}, 1]$ be a given parameter. Let $B = \min\{\eta N D, \ \inf\{b \ge D : \ \eta\beta_b b^2 \ge \epsilon_b D\}\}$. Algorithm 1 is an online learning algorithm for span($\mathcal{F}$) and losses in $\mathcal{C}$ with the following regret bound for any $f \in$ span($\mathcal{F}$):*

$$R'_f(T) \ \le \ \left(1 - \frac{\eta}{\|f\|_1}\right)^N \Delta_0 + 3\eta\beta_B B^2 \|f\|_1 T + L_B \|f\|_1 R(T) + 2L_B B \|f\|_1 \sqrt{T},$$

where $\Delta_0 := \sum_{t=1}^{T} \ell_t(\mathbf{0}) - \ell_t(f(\mathbf{x}_t))$.

[3] It suffices to compute upper bounds on these parameters.

Algorithm 1 Online Gradient Boosting for span($\mathcal{F}$)

```
Require: Number of weak learners $N$, step size parameter $\eta \in [\frac{1}{N}, 1]$,
Let $B = \min\{\eta N D,\ \inf\{b \geq D :\ \eta\beta_b b^2 \geq \epsilon_b D\}\}$.
Maintain $N$ copies of the algorithm $\mathcal{A}$, denoted $\mathcal{A}^i$ for $i = 1, 2, \ldots, N$.
For each $i$, initialize $\sigma^i_1 = 0$.
for $t = 1$ to $T$ do
    Receive example $\mathbf{x}_t$.
    Define $\mathbf{y}^0_t = \mathbf{0}$.
    for $i = 1$ to $N$ do
        Define $\mathbf{y}^i_t = \Pi_B((1 - \sigma^i_t\eta)\mathbf{y}^{i-1}_t + \eta\mathcal{A}^i(\mathbf{x}_t))$.
    end for
    Predict $\mathbf{y}_t = \mathbf{y}^N_t$.
    Obtain loss function $\ell_t$ and suffer loss $\ell_t(\mathbf{y}_t)$.
    for $i = 1$ to $N$ do
        Pass loss function $\ell^i_t(\mathbf{y}) = \frac{1}{L_B}\nabla\ell_t(\mathbf{y}^{i-1}_t) \cdot \mathbf{y}$ to $\mathcal{A}^i$.
        Set $\sigma^i_{t+1} = \max\{\min\{\sigma^i_t + \alpha_t\nabla\ell_t(\mathbf{y}^{i-1}_t) \cdot \mathbf{y}^{i-1}_t), 1\}, 0\}$, where $\alpha_t = \frac{1}{L_B B\sqrt{t}}$.
    end for
end for
```

The regret bound in this theorem depends on several parameters such as B, β_B and L_B. In applications of the algorithm for 1-dimensional regression with commonly used loss functions, however, these parameters are essentially modest constants; see Section 4 for calculations of the parameters for various loss functions. Furthermore, if η is appropriately set (e.g. $\eta = (\log N)/N$), then the average regret $R'_f(T)/T$ clearly converges to 0 as $N \to \infty$ and $T \to \infty$.

We also present a simpler boosting algorithm, Algorithm 2, that competes with CH($\mathcal{F}$). This algorithm has an optimal (up to constant factors) regret bound as given in the following theorem, proved in Section 3.2. The upper bound in this theorem is proved along the lines of the Frank-Wolfe [8] algorithm, and the lower bound using information-theoretic arguments.

Theorem 2. *Algorithm 2 is an online learning algorithm for CH($\mathcal{F}$) for losses in $\mathcal{C}$ with the regret bound*

$$R'(T) \ \leq \ \frac{8\beta_D D^2}{N}T + L_D R(T).$$

Furthermore, the dependence of this regret bound on N is optimal up to constant factors.

The dependence of the regret bound on $R(T)$ is unimprovable without additional assumptions: otherwise, Algorithm 2 will be an online linear learning algorithm over $\mathcal{F}$ with better than $R(T)$ regret.

Using a deterministic base online linear learning algorithm. If the base online linear learning algorithm $\mathcal{A}$ is deterministic, then our results can be improved, because our online boosting algorithms are also deterministic, and using a standard simple reduction, we can now allow $\mathcal{C}$ to be any set of convex functions (smooth or not) with a computable Lipschitz constant L_b over the domain $\mathbb{B}^d(b)$ for any $b > 0$.

This reduction converts arbitrary convex loss functions into linear functions: viz. if $\mathbf{y}_t$ is the output of the online boosting algorithm, then the loss function provided to the boosting algorithm

Algorithm 2 Online Gradient Boosting for CH($\mathcal{F}$)

```
Maintain $N$ copies of the algorithm $\mathcal{A}$, denoted $\mathcal{A}^1, \mathcal{A}^2, \ldots, \mathcal{A}^N$, and let $\eta_i = \frac{2}{i+1}$ for $i = 1, 2, \ldots, N$.
for $t = 1$ to $T$ do
    Receive example $\mathbf{x}_t$.
    Define $\mathbf{y}_t^0 = \mathbf{0}$.
    for $i = 1$ to $N$ do
        Define $\mathbf{y}_t^i = (1 - \eta_i)\mathbf{y}_t^{i-1} + \eta_i \mathcal{A}^i(\mathbf{x}_t)$.
    end for
    Predict $\mathbf{y}_t = \mathbf{y}_t^N$.
    Obtain loss function $\ell_t$ and suffer loss $\ell_t(\mathbf{y}_t)$.
    for $i = 1$ to $N$ do
        Pass loss function $\ell_t^i(\mathbf{y}) = \frac{1}{L_D}\nabla\ell_t(\mathbf{y}_t^{i-1}) \cdot \mathbf{y}$ to $\mathcal{A}^i$.
    end for
end for
```

as feedback is the linear function $\ell'_t(\mathbf{y}) = \nabla\ell_t(\mathbf{y}_t) \cdot \mathbf{y}$. This reduction immediately implies that the base online linear learning algorithm $\mathcal{A}$, when fed loss functions $\frac{1}{L_D}\ell'_t$, is already an online learning algorithm for CH($\mathcal{F}$) with losses in $\mathcal{C}$ with the regret bound $R'(T) \leq L_D R(T)$.

As for competing with span($\mathcal{F}$), since linear loss functions are 0-smooth, we obtain the following easy corollary of Theorem 1:

Corollary 1. *Let $\eta \in [\frac{1}{N}, 1]$ be a given parameter, and set $B = \eta N D$. Algorithm 1 is an online learning algorithm for span($\mathcal{F}$) for losses in $\mathcal{C}$ with the following regret bound for any $f \in$ span($\mathcal{F}$):*

$$R'_f(T) \;\leq\; \left(1 - \frac{\eta}{\|f\|_1}\right)^N \Delta_0 + L_B\|f\|_1 R(T) + 2L_B B\|f\|_1\sqrt{T},$$

where $\Delta_0 := \sum_{t=1}^T \ell_t(\mathbf{0}) - \ell_t(f(\mathbf{x}_t))$.

3 Analysis

In this section we provide rigorous analysis for our algorithms.

3.1 Analysis of Algorithm 1

Proof of Theorem 1. Let $f = \sum_{g \in S} w_g g$, for some finite subset S of $\mathcal{F}$, where $w_g \in \mathbb{R}$. Since $\mathcal{F}$ is symmetric, we may assume that all $w_g \geq 0$, and let $W := \sum_g w_g$. Furthermore, we may assume that $\mathbf{0} \in S$ with weight $w_{\mathbf{0}} = \max\{1 - \sum_{g \in S,\ g \neq \mathbf{0}} w_g, 0\}$, so that $W \geq 1$. Note that $\|f\|_1$ is exactly the infimum of W over all such ways of expressing f as a finite weighted sum of functions in $\mathcal{F}$. We now prove that bound stated in the theorem holds with $\|f\|_1$ replaced by W; the theorem then follows simply by taking the infimum of the bound over all such ways of expressing f.

Now, for each $i \in [N]$, the update in line 14 of Algorithm 1 is exactly online gradient descent [25] on the domain $[0, 1]$ with linear loss functions $\sigma \mapsto -\nabla\ell_t(\mathbf{y}_t^{i-1}) \cdot \mathbf{y}_t^{i-1}\sigma$. Note that the derivative of this loss function is bounded as follows: $|-\nabla \ell_t(\mathbf{y}_t^{i-1}) \cdot \mathbf{y}_t^{i-1}| \leq L_B B$. Since $\frac{1}{W} \in [0,1]$, the standard analysis of online gradient descent then implies that the sequence σ_t^i for $t = 1, 2, \ldots, T$ satisfies

$$\sum_{t=1}^{T} -\nabla \ell_t(\mathbf{y}_t^{i-1}) \cdot \mathbf{y}_t^{i-1} \sigma_t^i \ \leq\ \sum_{t=1}^{T} -\nabla \ell_t(\mathbf{y}_t^{i-1}) \cdot \mathbf{y}_t^{i-1} \frac{1}{W} + 2 L_B B \sqrt{T}. \tag{2}$$

Next, since $f = \sum_{g \in S} w_g g$ with $w_g \geq 0$, we have

$$\frac{1}{W} \sum_{t=1}^{T} \nabla \ell_t(\mathbf{y}_t^i) \cdot f(\mathbf{x}_t) = \frac{1}{\sum_{g \in S} w_g} \sum_{t=1}^{T} \sum_{g \in S} w_g \nabla \ell_t(\mathbf{y}_t^i) \cdot g(\mathbf{x}_t) \ \geq\ \min_{g \in S} \sum_{t=1}^{T} \nabla \ell_t(\mathbf{y}_t^i) \cdot g(\mathbf{x}_t). \tag{3}$$

Let $g^\star \in \arg\min_{g \in S} \sum_{t=1}^{T} \nabla \ell_t(\mathbf{y}_t^i) \cdot g(\mathbf{x}_t)$. Since $\mathcal{A}^i$ is an online learning algorithm for $\mathcal{F}$ with regret bound $R(\cdot)$ for the 1-Lipschitz linear loss functions $\ell_t^i(\mathbf{y}) = \frac{1}{L_B} \nabla \ell_t(\mathbf{y}_t^{i-1}) \cdot \mathbf{y}$, and $g^\star \in \mathcal{F}$, multiplying the regret bound (1) by L_B we have

$$\sum_{t=1}^{T} \nabla \ell_t(\mathbf{y}_t^{i-1}) \cdot \mathcal{A}^i(\mathbf{x}_t) \leq \sum_{t=1}^{T} \nabla \ell_t(\mathbf{y}_t^{i-1}) \cdot g^\star(\mathbf{x}_t) + L_B R(T) \leq \frac{1}{W} \sum_{t=1}^{T} \nabla \ell_t(\mathbf{y}_t^{i-1}) \cdot f(\mathbf{x}_t) + L_B R(T) \tag{4}$$

by (3). Now, we analyze how much excess loss is potentially introduced due to the projection in line 8. First, note that if $B = \eta N D$, then the projection has no effect since $(1 - \sigma_t^i \eta)\mathbf{y}_t^{i-1} + \eta \mathcal{A}^i(\mathbf{x}_t) \in \mathbb{B}^d(B)$, and in this case $\ell_t(\mathbf{y}_t^i) \ = \ \ell_t((1 - \sigma_t^i \eta)\mathbf{y}_t^{i-1} + \eta \mathcal{A}^i(\mathbf{x}_t))$. If $B < \eta N D$, then by the definition of B, $\eta \beta_B B^2 \geq \epsilon_B D$, and since $(1 - \sigma_t^i \eta)\mathbf{y}_t^{i-1} \in \mathbb{B}^d(B)$ and $\|\eta \mathcal{A}^i(\mathbf{x}_t))\| \leq \eta D$, and we have

$$\ell_t(\mathbf{y}_t^i) \ = \ \ell_t(\Pi_B((1 - \sigma_t^i \eta)\mathbf{y}_t^{i-1} + \eta \mathcal{A}^i(\mathbf{x}_t))) \ \leq \ \ell_t((1 - \sigma_t^i \eta)\mathbf{y}_t^{i-1} + \eta \mathcal{A}^i(\mathbf{x}_t)) + \eta \epsilon_B D.$$

In either case, we have

$$\ell_t(\mathbf{y}_t^i) \ \leq \ \ell_t((1 - \sigma_t^i \eta)\mathbf{y}_t^{i-1} + \eta \mathcal{A}^i(\mathbf{x}_t)) + \eta^2 \beta_B B^2. \tag{5}$$

We now move to the main part of the analysis. Define for $i = 0, 1, 2, \ldots, N$, $\Delta_i := \sum_{t=1}^{T} \ell_t(\mathbf{y}_t^i) - \ell_t(f(\mathbf{x}_t))$. We have

$$\begin{aligned}
\Delta_i \ &\leq \ \left[\sum_{t=1}^{T} \ell_t((1 - \sigma_t^i \eta)\mathbf{y}_t^{i-1} + \eta \mathcal{A}^i(\mathbf{x}_t)) - \ell_t(f(\mathbf{x}_t))\right] + \eta^2 \beta_B B^2 T \\
&\leq \ \Delta_{i-1} + \left[\sum_{t=1}^{T} \eta \nabla \ell_t(\mathbf{y}_t^{i-1}) \cdot (\mathcal{A}^i(\mathbf{x}_t) - \sigma_t^i \mathbf{y}_t^{i-1}) + \frac{\beta_B \eta^2}{2} \|\mathcal{A}^i(\mathbf{x}_t) - \sigma_t^i \mathbf{y}_t^{i-1}\|^2\right] + \eta^2 \beta_B B^2 T \\
&\quad \text{(by } \beta_B\text{-smoothness)} \\
&\leq \ \Delta_{i-1} \ + \left[\sum_{t=1}^{T} \frac{\eta}{W} \nabla \ell_t(\mathbf{y}_t^{i-1}) \cdot (f(\mathbf{x}_t) - \mathbf{y}_t^{i-1})\right] + 3\eta^2 \beta_B B^2 T + \eta L_B R(T) + 2\eta L_B B \sqrt{T} \\
&\quad \text{(by (2), (4) and the fact that } \|\mathcal{A}^i(\mathbf{x}_t) - \sigma_t^i \mathbf{y}_t^{i-1}\| \leq D + B \leq 2B) \\
&\leq \ \left(1 - \frac{\eta}{W}\right) \Delta_{i-1} + 3\eta^2 \beta_B B^2 T + \eta L_B R(T) + 2\eta L_B B \sqrt{T},
\end{aligned}$$

since, by convexity of ℓ_t we have $\ell_t(\mathbf{y}_t^{i-1}) + \nabla\ell(\mathbf{y}_t^{i-1}) \cdot (f(\mathbf{x}_t) - \mathbf{y}_t^{i-1}) \leq \ell_t(f(\mathbf{x}_t))$. Applying the above bound iteratively, we get

$$\begin{aligned}
\Delta_N &\leq \left(1 - \frac{\eta}{W}\right)^N \Delta_0 + \sum_{i=1}^{N} \left(1 - \frac{\eta}{W}\right)^{i-1} \cdot (3\eta^2\beta_B B^2 T + \eta L_B R(T) + 2\eta L_B B\sqrt{T}) \\
&\leq \left(1 - \frac{\eta}{W}\right)^N \Delta_0 + 3\eta\beta_B B^2 W T + L_B W R(T) + 2L_B B W\sqrt{T}.
\end{aligned}$$

This completes the proof. □

3.2 Analysis of Algorithm 2

Proof of Theorem 2. We prove the stated regret bound below. The lower bound statement follows directly from Theorem 3. First, note that for any $i = 1, 2, \ldots, N$, since ℓ_t^i is a linear function, we have

$$\inf_{f \in \mathrm{CH}(\mathcal{F})} \sum_{t=1}^{T} \ell_t^i(f(\mathbf{x}_t)) = \inf_{f \in \mathcal{F}} \sum_{t=1}^{T} \ell_t^i(f(\mathbf{x}_t)).$$

Let f be any function in $\mathrm{CH}(\mathcal{F})$. The equality above and the fact that $\mathcal{A}^i$ is an online learning algorithm for $\mathcal{F}$ with regret bound $R(\cdot)$ for the 1-Lipschitz linear loss functions $\ell_t^i(\mathbf{y}) = \frac{1}{L_D}\nabla\ell_t(\mathbf{y}_t^{i-1}) \cdot \mathbf{y}$ imply that

$$\sum_{t=1}^{T} \frac{1}{L_D} \nabla\ell_t(\mathbf{y}_t^{i-1}) \cdot \mathcal{A}^i(\mathbf{x}_t) \leq \sum_{t=1}^{T} \frac{1}{L_D} \nabla\ell_t(\mathbf{y}_t^{i-1}) \cdot f(\mathbf{x}_t) + R(T). \tag{6}$$

Now define, for $i = 0, 1, 2, \ldots, N$, $\Delta_i = \sum_{t=1}^{T} \ell_t(\mathbf{y}_t^i) - \ell_t(f(\mathbf{x}_t))$. We have

$$\begin{aligned}
\Delta_i &= \sum_{t=1}^{T} \ell_t(\mathbf{y}_t^{i-1} + \eta_i(\mathcal{A}^i(\mathbf{x}_t) - \mathbf{y}_t^{i-1})) - \ell_t(f(\mathbf{x}_t)) \\
&\leq \sum_{t=1}^{T} \ell_t(\mathbf{y}_t^{i-1}) - \ell_t(f(\mathbf{x}_t)) + \eta_i \nabla\ell_t(\mathbf{y}_t^{i-1}) \cdot (\mathcal{A}^i(\mathbf{x}_t) - \mathbf{y}_t^{i-1}) + \frac{\eta_i^2 \beta_D}{2} \|\mathcal{A}^i(\mathbf{x}_t) - \mathbf{y}_t^{i-1}\|^2 \\
&\quad \text{(by } \beta_D\text{-smoothness of } \mathcal{C}) \\
&\leq \left[\sum_{t=1}^{T} \ell_t(\mathbf{y}_t^{i-1}) - \ell_t(f(\mathbf{x}_t)) + \eta_i \nabla\ell_t(\mathbf{y}_t^{i-1}) \cdot (f(\mathbf{x}_t) - \mathbf{y}_t^{i-1}) + 2\eta_i^2 \beta_D D^2\right] + \eta_i L_D R(T) \\
&\quad \text{(by (6) and using the bound } \|\mathcal{A}^i(\mathbf{x}_t) - \mathbf{y}_t^{i-1}\| \leq 2D) \\
&\leq \left[\sum_{t=1}^{T} \ell_t(\mathbf{y}_t^{i-1}) - \ell_t(f(\mathbf{x}_t)) - \eta_i(\ell_t(\mathbf{y}_t^{i-1}) - \ell_t(f(\mathbf{x}_t))) + 2\eta_i^2 \beta_D D^2\right] + \eta_i L_D R(T) \\
&\quad \big(\text{by convexity, } \ell_t(\mathbf{y}_t^{i-1}) + \nabla\ell(\mathbf{y}_t^{i-1}) \cdot (f(\mathbf{x}_t) - \mathbf{y}_t^{i-1}) \leq \ell_t(f(\mathbf{x}_t))\big) \\
&\leq (1 - \eta_i)\Delta_{i-1} + 2\eta_i^2 \beta_D D^2 T + \eta_i L_D R(T).
\end{aligned}$$

For $i = 1$, since $\eta_1 = 1$, the above bound implies that $\Delta_1 \leq 2\beta_D D^2 T + L_D R(T)$. Starting from this base case, an easy induction on $i \geq 1$ proves that $\Delta_i \leq \frac{8\beta_D D^2}{i} T + L_D R(T)$. Applying this bound for $i = N$ completes the proof. □

We now show that the dependence of the regret bound of Algorithm 2 on the parameter N is optimal up to constant factors.

Theorem 3. *Let N be any specified bound on the total number of calls in each iteration to all copies of the base online linear learning algorithm. Then there is a setting of 1-dimensional prediction with a 1-bounded comparator function class $\mathcal{F}$, an online linear optimization algorithm $\mathcal{A}$ over $\mathcal{F}$, and a class $\mathcal{C}$ of loss functions that is 1-smooth on $\mathbb{R}$ such that any online boosting algorithm for CH($\mathcal{F}$) with losses in $\mathcal{C}$ respecting the bound N has regret at least $\Omega(\frac{T}{N})$.*

Proof. Consider the following construction. At a high level, the setting is 1-dimensional regression with $\mathcal{C}$ corresponding to squared loss. The domain $\mathcal{X} = \mathbb{N}$ and true labels of examples are in $[0, 1]$.

Define $p_1 = \frac{1}{2} + \epsilon$ and $p_2 = \frac{1}{2} - \epsilon$, where $\epsilon = \frac{1}{10\sqrt{N}}$, and let D_1 and D_2 be two distributions over $\{0, 1\}^N$ where each bit is Bernoulli random variable with parameter p_1 and p_2 respectively, chosen independently of the other bits. Consider a sequence of examples $(\mathbf{x}_t, y_t^\star) \in \mathbb{N} \times [0, 1]$ generated as follows: $\mathbf{x}_t = t$, and the label $y_t^\star$ is chosen from $\{p_1, p_2\}$ uniformly at random in each round.

Let for $c = \frac{1}{4000}$. The function class $\mathcal{F}$ consists of a large number, $M = \frac{1}{c}N$, of functions f_i, $i \in [M]$. For each t and i, we set $f_i(\mathbf{x}_t) = 1$ w.p. $y_t^\star$, and 0 w.p. $1 - y_t^\star$, independently of all other values of t and i.

The base online linear learning algorithm $\mathcal{A}$ is simply Hedge over the M functions. In each round, the Hedge algorithm selects one of the M functions in $\mathcal{F}$ and uses that to predict the label, and for any sequence of T examples, with high probability, incurs regret $R(T) = O(\sqrt{\log(M)T})$.

We set $\mathcal{C}$ to be set of squared loss functions, i.e. functions of the form $\ell(y) = \frac{1}{2}(y - y^\star)^2$ for $y^\star \in [0, 1]$. Note that these loss functions are 1-smooth and $D = 1$. In round t, the loss function is $\ell_t(y) = \frac{1}{2}(y - y_t^\star)^2$.

Consider the function $\bar{f} = \frac{1}{M}\sum_{i \in [M]} f_i$, which is in CH($\mathcal{F}$). Given any input sequence $(\mathbf{x}_t, y_t^\star)$ for $t = 1, 2, \ldots, T$ it is easy to calculate that $\mathbb{E}[\frac{1}{2}(\bar{f}(\mathbf{x}_t) - y_t^\star)^2] = \frac{y_t^\star(1-y_t^\star)}{2M} \leq \frac{1}{2M}$, and since the examples and predictions of functions on the examples are independent across iterations, a simple application of the multiplicative Chernoff bound implies that if $T \geq 12M$, then with probability at least 0.9, we have $\sum_{t=1}^T \frac{1}{2}(\bar{f}(\mathbf{x}_t) - y_t^\star)^2 \leq \frac{T}{M}$.

Now suppose there is an online boosting algorithm making at most N calls total to all copies of $\mathcal{A}$ in each iteration, that for any large enough T and for any sequence $(\mathbf{x}_t, y_t^\star)$ for $t = 1, 2, \ldots, T$, outputs predictions y_t such that with high probability, say at least 0.9, we have $\sum_{t=1}^T \frac{1}{2}(y_t - y_t^\star)^2 \leq \sum_{t=1}^T \frac{1}{2}(\bar{f}(\mathbf{x}_t) - y_t^\star)^2 + \frac{cT}{N}$. Then by a union bound, with probability at least 0.8, we have $\sum_{t=1}^T \frac{1}{2}(y_t - y_t^\star)^2 \leq \frac{cT}{N} + \frac{T}{M} \leq \frac{2cT}{N}$. By Markov's inequality and a union bound, with probability at least 0.7, for a uniform random time $\tau \in [T]$, we have

$$\frac{1}{2}(y_\tau - y_\tau^\star)^2 \;\leq\; \frac{20c}{N} \;=\; \frac{\epsilon^2}{2}, \tag{7}$$

or in other words, y_τ is on the same side of $\frac{1}{2}$ as $y_\tau^\star$, and thus can be used to identify $y_\tau^\star$. In the rest of the proof, we will use this fact, along with fact the total variation distance between D_1 and D_2, denoted $d_{\mathrm{TV}}(D_1, D_2)$, is small, to derive a contradiction.

Define the random variable $Y : \{0, 1\}^N \to \mathbb{R}$ as follows. For any bit string $s = \langle s_1, s_2, \ldots, s_N \rangle \in \{0, 1\}^N$, choose a random round $\tau \in [T]$, and simulate the online boosting process until round $\tau - 1$ by sampling $y_t^\star$'s and the outputs of $f_i(\mathbf{x}_t)$ for all $t \leq \tau - 1$ and $i \in [M]$ from the appropriate distributions. In round τ, let $f_{i_1}, f_{i_2}, \ldots, f_{i_N}$ be the functions that are obtained from the at most N

calls to copies of $\mathcal{A}$ (there could be repetitions). Assign $f_{i_j}(\mathbf{x}_\tau) = s_j$ for $j \in [N]$ (being careful with repeated functions and repeating outputs appropriately), and run the booster with these outputs to obtain y_τ, and set $Y(s) = y_\tau$. Let $\Pr[\cdot]$ denotes probability of events in this process for generating $Y(s)$ given s.

Let $\mathbb{E}_1[X(s)]$ and $\mathbb{E}_2[X(s)]$ denote expectation of a random variable $X : \{0,1\}^N \to \mathbb{R}$ when s is drawn from D_1 and D_2 respectively, and let $\mathbb{E}_0[X(I,s)]$ denote expectation of a random variable $X : \{1,2\} \times \{0,1\}^N \to \mathbb{R}$ when I is chosen from $\{1,2\}$ uniformly at random and then s is sampled from D_I. The above analysis (inequality (7)) implies that

$$0.7 \ \leq \ \mathbb{E}_0[\Pr[|Y(s) - p_I| \leq \epsilon]] \ = \ \tfrac{1}{2}\mathbb{E}_1[\Pr[|Y(s) - p_1| \leq \epsilon]] + \tfrac{1}{2}\mathbb{E}_2[\Pr[|Y(s) - p_2| \leq \epsilon]].$$

Now define a random variable $X : \{0,1\}^N \to \mathbb{R}$ as $X(s) = \Pr[Y(s) \geq \frac{1}{2}]$. Since

$$\Pr[Y(s) \geq \tfrac{1}{2}] \ \geq \ \Pr[|Y(s) - p_1| \leq \epsilon] \ \text{ and } \ 1 - \Pr[Y(s) \geq \tfrac{1}{2}] \ \geq \ \Pr[|Y(s) - p_2| \leq \epsilon],$$

we conclude, using the above bound, that $\mathbb{E}_1[X(s)] - \mathbb{E}_2[X(s)] \geq 0.4$. This is a contradiction, since because $X(s) \in [0,1]$, we have

$$\mathbb{E}_1[X(s)] - \mathbb{E}_2[X(s)] \ \leq \ d_{\mathrm{TV}}(D_1, D_2) \ < \ 4\sqrt{\epsilon^2 N} \ = \ 0.4,$$

where the bound on $d_{\mathrm{TV}}(D_1, D_2)$ is standard, for e.g. see [15]. This gives us the desired contradiction. □

The above result can be easily extended to any given parameters β and D so that the $\mathcal{F}$ is D-bounded and $\mathcal{C}$ is β-smooth on $\mathbb{R}$, giving a lower bound of $\Omega(\frac{\beta D^2 T}{N})$ on the regret of an online boosting algorithm for $\mathrm{CH}(\mathcal{F})$ with losses in $\mathcal{C}$: we simply scale all function and label values by D, and consider the loss functions $\ell(y, y^\star) = \frac{\beta}{2}(y - y^\star)^2$. If there were an online boosting algorithm for $\mathrm{CH}(\mathcal{F})$ with these loss functions with regret $o(\frac{\beta D^2 T}{N})$, then by scaling down the predictions by D, we obtain an online boosting algorithm for exactly the setting in the proof of Theorem 3 with a regret bound of $o(\frac{T}{N})$, which is a contradiction.

4 The parameters for several basic loss functions

In this section we consider the application of our results to 1-dimensional regression, where we assume, for normalization, that the true labels of the examples and the predictions of the functions in the class $\mathcal{F}$ are in $[-1,1]$. In this case $\|\cdot\|$ denotes the absolute value norm. Thus, in each round, the adversary chooses a labeled data point $(\mathbf{x}_t, y_t^\star) \in \mathcal{X} \times [-1,1]$, and the loss for the prediction $y_t \in [-1,1]$ is given by $\ell_t(y_t) = \ell(y_t^\star, y_t)$ where $\ell(\cdot,\cdot)$ is a fixed loss function that is convex in the second argument. Note that $D = 1$ in this setting. We give examples of several such loss functions below, and compute the parameters L_b, β_b and ϵ_b for every $b > 0$, as well as B from Theorem 1.

1. Linear loss: $\ell(y^\star, y) = -y^\star y$. We have $L_b = 1$, $\beta_b = 0$, $\epsilon_b = 1$, and $B = \eta N$.
2. p-norm loss, for some $p \geq 2$: $\ell(y^\star, y) = |y^\star - y|^p$. We have $L_b = p(b+1)^{p-1}$, $\beta_b = p(p-1)(b+1)^{p-2}$, $\epsilon_b = \max\{p(1-b)^{p-1}, 0\}$, and $B = 1$.
3. Modified least squares: $\ell(y^\star, y) = \frac{1}{2}\max\{1 - y^\star y, 0\}^2$. We have $L_b = b + 1$, $\beta_b = 1$, $\epsilon_b = \max\{1 - b, 0\}$, and $B = 1$.
4. Logistic loss: $\ell(y^\star, y) = \ln(1 + \exp(-y^\star y))$. We have $L_b = \frac{\exp(b)}{1+\exp(b)}$, $\beta_b = \frac{1}{4}$, $\epsilon_b = \frac{\exp(-b)}{1+\exp(-b)}$, and $B = \min\{\eta N, \ln(4/\eta)\}$.

5 Variants of the boosting algorithms

Our boosting algorithms and the analysis are considerably flexible: it is easy to modify the algorithms to work with a different (and perhaps more natural) kind of base learner which does greedy fitting, or incorporate a scaling of the base functions which improves performance. Also, when specialized to the batch setting, our algorithms provide better convergence rates than previous work.

5.1 Fitting to actual loss functions

The choice of an online *linear* learning algorithm over the base function class in our algorithms was made to ease the analysis. In practice, it is more common to have an online algorithm which produce predictions with comparable accuracy to the best function in hindsight for the *actual* sequence of loss functions. In particular, a common heuristic in boosting algorithms such as the original gradient boosting algorithm by Friedman [10] or the matching pursuit algorithm of Mallat and Zhang [18] is to build a linear combination of base functions by iteratively augmenting the current linear combination by greedily choosing a base function and a step size for it that minimizes the loss with respect to the residual label. Indeed, the boosting algorithm of Zhang and Yu [24] also uses this kind of greedy fitting algorithm as the base learner.

In the online setting, we can model greedy fitting as follows. We first fix a step size $\alpha \geq 0$ in advance. Then, in each round t, the base learner $\mathcal{A}$ receives not only the example $\mathbf{x}_t$, but also an *offset* $\mathbf{y}'_t \in \mathbb{R}^d$ for the prediction, and produces a prediction $\mathcal{A}(\mathbf{x}_t) \in \mathbb{R}^d$, after which it receives the loss function ℓ_t and suffers loss $\ell_t(\mathbf{y}'_t + \alpha\mathcal{A}(\mathbf{x}_t))$. The predictions of $\mathcal{A}$ satisfy

$$\sum_{t=1}^{T} \ell_t(\mathbf{y}'_t + \alpha\mathcal{A}(\mathbf{x}_t)) \;\leq\; \inf_{f\in\mathcal{F}} \sum_{t=1}^{T} \ell_t(\mathbf{y}'_t + \alpha f(\mathbf{x}_t)) + R(T),$$

where R is the regret. We now describe how our algorithms can be made to work with this kind of base learner as well.

Assume that for some known parameter $B > 0$, we have $\|\mathbf{y}'_t\| \leq B$, for all t. Let $B' = B + \alpha D$, and assume that the loss functions ℓ_t are $L_{B'}$ Lipschitz and $\beta_{B'}$ smooth on $\mathbb{B}^d(B')$. Then using the convexity and smoothness of the loss functions, we have $\ell_t(\mathbf{y}'_t + \alpha\mathcal{A}(\mathbf{x}_t)) \geq \ell_t(\mathbf{y}'_t) + \alpha\nabla\ell_t(\mathbf{y}'_t)\cdot\mathcal{A}(\mathbf{x}_t)$ and $\ell_t(\mathbf{y}'_t + \alpha f(\mathbf{x}_t)) \leq \ell_t(\mathbf{y}'_t) + \alpha\nabla\ell_t(\mathbf{y}'_t)\cdot f(\mathbf{x}_t) + \frac{\beta_{B'}\alpha^2}{2}\|f(\mathbf{x}_t)\|^2$. Plugging these bounds into the above regret bound we get, for any $f \in \mathcal{F}$,

$$\sum_{t=1}^{T} \nabla\ell_t(\mathbf{y}'_t)\cdot\mathcal{A}(\mathbf{x}_t) \;\leq\; \sum_{t=1}^{T}\left(\nabla\ell_t(\mathbf{y}'_t)\cdot f(\mathbf{x}_t) + \frac{\beta_{B'}}{2}\alpha\|f(\mathbf{x}_t)\|^2\right) + \frac{1}{\alpha}R(T).$$

Since $\|f(\mathbf{x}_t)\| \leq D$, setting $\alpha = \sqrt{\frac{2R(T)}{\beta_{B'}D^2T}}$, we conclude that

$$\sum_{t=1}^{T} \nabla\ell_t(\mathbf{y}'_t)\cdot\mathcal{A}(\mathbf{x}_t) \;\leq\; \sum_{t=1}^{T} \nabla\ell_t(\mathbf{y}'_t)\cdot f(\mathbf{x}_t) + \sqrt{2\beta_{B'}D^2TR(T)}. \tag{8}$$

This regret bound is sublinear in T if $R(T)$ is sublinear. We can obtain a better regret bound if we assume that $R(T)$ scales linearly with α: this is a natural assumption since the functions

$\ell_t(\mathbf{y}'_t + \alpha\mathbf{y})$ are $\alpha L_{B'}$ Lipschitz in the prediction $\mathbf{y}$. In this case, the regret bound $R(T) = \alpha R'(T)$ for some fixed $R' : \mathbb{N} \to \mathbb{R}_+$ indepedent of α, and we can choose $\alpha = \frac{2R'(T)}{\beta_{B'} D^2 T}$ so that

$$\sum_{t=1}^{T} \nabla\ell_t(\mathbf{y}'_t) \cdot \mathcal{A}(\mathbf{x}_t) \;\leq\; \sum_{t=1}^{T} \nabla\ell_t(\mathbf{y}'_t) \cdot f(\mathbf{x}_t) + 2R'(T). \tag{9}$$

Either the bound (8) or (9) suffices for the analysis of our boosting algorithms to go through: to use this kind of base learner $\mathcal{A}$, we again keep N copies $\mathcal{A}^1, \mathcal{A}^2, \ldots, \mathcal{A}^N$ with a suitably chosen step size α, and simply change line 11 of Algorithm 2 and line 13 of Algorithm 1 to pass the offset $\mathbf{y}'_t = \mathbf{y}_t^{i-1}$ to $\mathcal{A}^i$.

5.2 Improving the regret bound via scaling

Given an online linear learning algorithm $\mathcal{A}$ over the function class $\mathcal{F}$ with regret R, then for any scaling parameter $\lambda > 0$, we trivially obtain an online linear learning algorithm, denoted $\lambda\mathcal{A}$, over a λ-scaling of $\mathcal{F}$, viz. $\lambda\mathcal{F} := \{\lambda f \mid f \in \mathcal{F}\}$, simply by multiplying the predictions of $\mathcal{A}$ by λ. The corresponding regret scales by λ as well, i.e. it becomes λR.

The performance of Algorithm 1 can be improved by using such an online linear learning algortihm over $\lambda\mathcal{F}$ for a suitably chosen scaling $\lambda \geq 1$ of the function class $\mathcal{F}$. Let $\|f\|'_1 = \max\{1, \frac{\|f\|_1}{\lambda}\}$ be the 1-norm of f measured with respect to $\lambda\mathcal{F}$, and $B' = \min\{\eta N\lambda D,\ \inf\{b \geq \lambda D :\ \eta\beta_b b^2 \geq \epsilon_b \lambda D\}\}$. Then we immediately get the following corollary of Theorem 1:

Corollary 2. *For any $f \in$ span$(\mathcal{F})$, let $\Delta_0 = \sum_{t=1}^{T} \ell_t(0) - \ell_t(f(\mathbf{x}_t))$. Algorithm 1, using $\lambda\mathcal{A}$ as the online linear algorithm over $\lambda\mathcal{F}$, is an online learning algorithm for span$(\mathcal{F})$ for losses in $\mathcal{C}$ with the following regret bound for any $f \in$ span$(\mathcal{F})$:*

$$R'_f(T) \leq \left(1 - \frac{\eta}{\|f\|'_1}\right)^N \Delta_0 + 3\eta\beta_{B'}{B'}^2\|f\|'_1 T + L_{B'}\|f\|'_1 \lambda R(T) + 2L_{B'}B'\|f\|'_1\sqrt{T}.$$

Choosing large values of λ implies that $\|f\|'_1$ can be significantly smaller than $\|f\|_1$. But B' becomes bigger than B, and correspondingly, the parameters $\beta_{B'}$ and $L_{B'}$ become bigger than β_B and L_B respectively. Also, the (lower order) dependence on the regret term $R(T)$ increases by a factor of λ.

However, it turns out (see Section 4) that in several common applications of the algorithm, B' can be set to be equal to B or the increase from B is a very slow growing function of λ, such as $\log(\lambda)$. In such situations choosing larger values of λ leads to improvement in the higher order terms of the regret bound, while making the lower order term (i.e. $L_{B'}\|f\|'_1\lambda R(T)$) worse; overall the regret bound can be improved by choosing a suitably large scaling factor λ to balance between the two.

5.3 Improvements for batch boosting

Our algorithmic technique can be used to improve convergence speed for batch boosting as well, in the setup considered by Zhang and Yu [24]. Since the focus of this paper is on online boosting, we give a high level comparison of the bounds here, making some simplifying assumptions to ease the technical details, using our notation as much as possible.

In the setup of Zhang and Yu [24], we have a base set of real valued functions $\mathcal{F}$, which we assume is symmetric and contains the zero function, $\mathbf{0}$. Then $\operatorname{span}(\mathcal{F})$ is a linear function space, and let $\|\cdot\|$ be some norm defined on $\operatorname{span}(\mathcal{F})$. For clarity of presentation, we assume that for any $f \in \mathcal{F}$, we have $\|f\| \leq 1$. This implies that for any $f \in \operatorname{span}(\mathcal{F})$, we have $\|f\| \leq \|f\|_1$.

The goal is to minimize a given convex functional $\ell : \operatorname{span}(\mathcal{F}) \to \mathbb{R}$ over its domain, $\operatorname{span}(\mathcal{F})$. We assume, for simplicity, that ℓ is β-smooth over $\operatorname{span}(\mathcal{F})$ under the norm $\|\cdot\|$, i.e. for any $f, f' \in \operatorname{span}(\mathcal{F})$, we have

$$\ell(f') \;\leq\; \ell(f) + \nabla\ell(f) \cdot (f' - f) + \frac{\beta}{2}\|f - f'\|^2.$$

Zhang and Yu [24] assume[4] that we have access to a base learning algorithm $\mathcal{A}$ that, given any $f \in \operatorname{span}(\mathcal{F})$ and a step size $\eta \geq 0$ can find a function $g \in \mathcal{F}$ that minimizes $\ell(f + \eta g)$. We denote the output of $\mathcal{A}$ by $\mathcal{A}(f, \eta)$.

Given such a base learning algorithm, and a sequence of step sizes $\eta_1, \eta_2, \ldots$, the boosting algorithm of Zhang and Yu [24] computes a sequence of functions $f_0, f_1, f_2, \ldots \in \operatorname{span}(\mathcal{F})$ via greedy fitting as follows: f_0 is set to $\mathbf{0}$, and for any $i \geq 1$,

$$f_i \;:=\; f_{i-1} + \eta_i \mathcal{A}(f_{i-1}, \eta_i).$$

Define $s_0 = 1$ and $s_i = s_{i-1} + \eta_i$ for any $i \geq 1$.

For any $f \in \operatorname{span}(f)$, for $i = 1, 2, \ldots$, let $\Delta_i = \ell(f_i) - \ell(f)$ denote the optimization errors of the function f_i. Zhang and Yu [24] prove that for any $N \in \mathbb{N}$, we have

$$\Delta_N \;\leq\; \frac{s_0 + \|f\|_1}{s_N + \|f\|_1}\Delta_0 + \sum_{i=1}^{N} \frac{s_i + \|f\|_1}{s_N + \|f\|_1} \cdot \frac{\beta}{2}\eta_i^2. \tag{10}$$

Using the techniques in this paper, we can define a different boosting algorithm which works as follows. Given the same sequence of step sizes $\eta_1, \eta_2, \ldots$ as above, we set $f_0 = \mathbf{0}$, and for any $i \geq 1$,

$$f_i \;:=\; (1 - \sigma_i \eta_i) f_{i-1} + \eta_i \mathcal{A}(f_{i-1}, \eta_i),$$

where

$$\sigma_i \;:=\; \begin{cases} 1 & \text{if } \nabla\ell(f_{i-1}) \cdot f_{i-1} \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

We can analyze this algorithm along the lines of the proof of Theorem 1. First, let $g_i = \mathcal{A}(f_{i-1}, \eta_i)$. Then for $g \in \mathcal{F}$, we have $\ell(f_{i-1} + \eta_i g_i) \leq \ell(f_{i-1} + \eta_i g)$, and by the convexity and β-smoothness of ℓ, we conclude that

$$\nabla\ell(f_{i-1}) \cdot g_i \;\leq\; \nabla\ell(f_{i-1}) \cdot g + \frac{\beta}{2}\eta_i.$$

Using this fact and following the proof of Theorem 1, we get the following bound on the optimization error $\Delta_i = \ell(f_i) - \ell(f)$ of the function f_i:

$$\Delta_N \;\leq\; \exp\left(-\frac{s_N - s_0}{\|f\|_1}\right)\Delta_0 + \sum_{i=1}^{N} \exp\left(-\frac{s_N - s_i}{\|f\|_1}\right) \cdot \frac{\beta}{2}\eta_i^2(s_i^2 + 1). \tag{11}$$

[4] This is a slight simplification of the base learning algorithm considered in [24], which also performs a search over the step size η. Also, the analysis in [24] allows some optimization error for the base learning algorithm; to simplify the comparison we assume this error is 0.

We can compare our bound (11) to the bound (10) of Zhang and Yu [24], by comparing corresponding terms in the bound. For each term, we can calculate how large s_N needs to be for the term to be reduced to less than some given bound ϵ. To reduce the first term to less than ϵ our algorithm needs $s_N \geq \|f\|_1 \log(\frac{\Delta_0}{\epsilon}) + s_0$, whereas the algorithm of Zhang and Yu [24] needs $s_N \geq (\frac{\Delta_0}{\epsilon})(s_0 + \|f\|_1) - \|f\|_1$. As for the second term, to reduce the i-th term in the sum to less than ϵ, our algorithm needs $s_N \geq \|f\|_1 \log(\frac{\beta\eta_i^2(s_i^2+1)}{2\epsilon}) + s_i$, whereas the algorithm of Zhang and Yu [24] needs $s_N \geq (\frac{\beta\eta_i^2}{2\epsilon})(s_i + \|f\|_1) - \|f\|_1$. Since in either case, the dependence on ϵ is $\log(\frac{1}{\epsilon})$ for our algorithm, whereas it is $\frac{1}{\epsilon}$ for the algorithm of Zhang and Yu [24], we conclude that our algorithm converges exponentially faster.

6 Experimental Results

Is it possible to boost in an online fashion in practice with real base learners? To study this question, we implemented and evaluated Algorithms 1 and 2 within the Vowpal Wabbit (VW) open source machine learning system [23]. The three online base learners used were VW's default linear learner (a variant of stochastic gradient descent), two-layer sigmoidal neural networks with 10 hidden units, and regression stumps.

Regression stumps were implemented by doing stochastic gradient descent on each individual feature, and predicting with the best-performing non-zero valued feature in the current example.

All experiments were done on a collection of 14 publically available regression and classification datasets, described in Section A. For all experiments, the target loss was squared loss. The only parameters tuned were the learning rate and the number of weak learners, as well as the step size parameter for Algorithm 1. Parameters were tuned based on progressive validation loss on half of the dataset; reported is propressive validation loss on the remaining half. Progressive validation is a standard online validation technique, where each training example is used for testing before it is used for updating the model [3].

The following table reports the average and the median, over the datasets, relative improvement in squared loss over the respective base learner. Detailed results can be found in Section A.

Base learner	Average relative improvement (%)		Median relative improvement (%)	
	Algorithm 1	Algorithm 2	Algorithm 1	Algorithm 2
SGD	1.65%	1.33%	0.03%	0.29%
Regression stumps	20.22%	15.9%	10.45%	13.69%
Neural networks	7.88%	0.72%	0.72%	0.33%

It is not surprising that boosting is much more effective for regression stumps, which is a weak base learner. Both SGD (stochastic gradient descent) and neural networks are already very strong learners.

7 Conclusions and Future Work

In this paper we generalized the theory of boosting for regression problems to the online setting and provided online boosting algorithms with theoretical convergence guarantees. Our algorithmic technique also improves convergence guarantees for batch boosting algorithms. We also provide experimental evidence that our boosting algorithms do improve prediction accuracy over commonly used base learners in practice, with greater improvements for weaker base learners. The main remaining open question is whether the boosting algorithm for competing with the span of the base functions is optimal in any sense, similar to our proof of optimality for the the boosting algorithm for competing with the convex hull of the base functions.

References

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

[2] Alina Beygelzimer, Satyen Kale, and Haipeng Luo. Optimal and adaptive algorithms for online boosting. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

[3] Avrim Blum, Adam Kalai, and John Langford. Beating the hold-out: Bounds for k-fold and progressive cross-validation. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, COLT '99, pages 203–208, 1999.

[4] Shang-Tse Chen, Hsuan-Tien Lin, and Chi-Jen Lu. An Online Boosting Algorithm with Theoretical Justifications. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

[5] Shang-Tse Chen, Hsuan-Tien Lin, and Chi-Jen Lu. Boosting with Online Binary Learners for the Multiclass Bandit Problem. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

[6] Michael Collins, Robert E. Schapire, and Yoram Singer. Logistic regression, AdaBoost and Bregman distances. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, 2000.

[7] Nigel Duffy and David Helmbold. Boosting methods for regression. *Machine Learning*, 47 (2/3):153–200, 2002.

[8] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Res. Logis. Quart.*, 3:95–110, 1956.

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

[10] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5), October 2001.

[11] Helmut Grabner and Horst Bischof. On-line boosting and vision. In *CVPR*, volume 1, pages 260–267, 2006.

[12] Helmut Grabner, Christian Leistner, and Horst Bischof. Semi-supervised on-line boosting for robust tracking. In *ECCV*, pages 234–247, 2008.

[13] Trevor Hastie and R. J Robet Tibshirani. *Generalized Additive Models.* Chapman and Hall, 1990.

[14] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer Verlag, 2001.

[15] Elad Hazan and Satyen Kale. Beyond the regret minimization barrier: optimal algorithms for stochastic strongly-convex optimization. *Journal of Machine Learning Research*, 15(1): 2489–2512, 2014.

[16] Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007.

[17] Xiaoming Liu and Ting Yu. Gradient feature selection for online boosting. In *ICCV*, pages 1–8, 2007.

[18] Stéphane G. Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, December 1993.

[19] Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Boosting algorithms as gradient descent. In *Advances in Neural Information Processing Systems 12*, 2000.

[20] Nikunj C. Oza and Stuart Russell. Online bagging and boosting. In *Eighth International Workshop on Artificial Intelligence and Statistics*, pages 105–112, 2001.

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

[22] Matus Telgarsky. Boosting with the logistic loss is consistent. In *Proceedings of the 26th Annual Conference on Learning Theory*, 2013.

[23] VW. URL `https://github.com/JohnLangford/vowpal_wabbit/`.

[24] Tong Zhang and Bin Yu. Boosting with early stopping: Convergence and consistency. *Annals of Statistics*, 33(4):1538–1579, 2005.

[25] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

A Description of Data Sets and Detailed Experimental Results

The datasets come from the UCI repository and various KDD Cup challenges. Below, d is the number of unique features in the dataset, and s is the average number of features per example.

Dataset	Number of instances	Total number of features	Average number of features per example	Task	Label range
a9a	48,841	123	14	classification	$[-1, 1]$
abalone	4,177	10	9	regression	$[1, 29]$
activity	165,632	20	18.5	classification	$[-1, 1]$
adult	48,842	105	12	classification	$[0, 1]$
bank	45,211	45	15	classification	$[-1, 1]$
cal_housing	20,640	9	9	regression	$[0, 1]$
casp	45,730	10	10	regression	$[0, 1]$
census	299,284	401	32	classification	$[-1, 1]$
covtype	581,011	54	12	classification	$[-1, 1]$
kddcup04 (phy)	50,000	74	32	classification	$[0, 1]$
letter	20,000	16	15.6	classification	$[-1, 1]$
shuttle	43,500	9	8	classification	$[-1, 1]$
slice	53,500	385	135	regression	$[0, 1]$
year	463,715	90	90	regression	$[0, 1]$

The following table provides online squared losses summarized in Section 6.

	SGD			Regression stumps			Neural Networks		
Dataset	Baseline	Alg 1	Alg 2	Baseline	Alg 1	Alg 2	Baseline	Alg 1	Alg 2
kddcup04/phy	0.7475	0.7466	0.7470	0.9201	0.7733	0.7924	0.7441	0.7480	0.7446
cal_housing	0.0094	0.0094	0.0104	0.0151	0.0138	0.0124	0.0096	0.0096	0.0107
casp	0.0632	0.0631	0.0630	0.0741	0.0741	0.0742	0.0639	0.0632	0.0631
a9a	0.4261	0.4283	0.4249	0.5749	0.5074	0.5758	0.4256	0.4266	0.4246
abalone	3.7263	3.7482	3.7154	6.7791	3.8273	4.2270	3.7380	3.7255	3.7212
activity	0.0334	0.0337	0.0316	0.4492	0.1454	0.3141	0.0192	0.0143	0.0186
adult	0.1055	0.1057	0.1056	0.1388	0.1261	0.1250	0.1081	0.1062	0.1081
bank	0.2971	0.2968	0.2973	0.3774	0.3240	0.3257	0.2962	0.2969	0.2969
census	0.1544	0.1545	0.1553	0.2073	0.1884	0.1789	0.1531	0.1531	0.1523
covtype	0.7256	0.7270	0.7286	0.7910	0.7986	0.7911	0.6807	0.6465	0.6757
letter	0.6441	0.5698	0.6108	0.7420	0.7087	0.7168	0.6542	0.5729	0.6108
shuttle	0.1616	0.1547	0.1577	0.8551	0.3678	0.4354	0.0760	0.0694	0.0802
slice	0.0076	0.0067	0.0065	0.0559	0.0362	0.0410	0.0054	0.0022	0.0044
year	0.0116	0.0119	0.0115	0.0152	0.0140	0.0141	0.0116	0.0119	0.0122