Accelerating Gradient Boosting Machine

Haihao Lu* Sai Praneeth Karimireddy† Natalia Ponomareva ‡ Vahab Mirrokni §

January, 2019

Abstract

Gradient Boosting Machine (GBM) [14] is an extremely powerful supervised learning algorithm that is widely used in practice. GBM routinely features as a leading algorithm in machine learning competitions such as Kaggle and the KDDCup [7]. In this work, we propose Accelerated Gradient Boosting Machine (AGBM) by incorporating Nesterov's acceleration techniques into the design of GBM. The difficulty in accelerating GBM lies in the fact that weak (inexact) learners are commonly used, and therefore the errors can accumulate in the momentum term. To overcome it, we design a "corrected pseudo residual" and fit best weak learner to this corrected pseudo residual, in order to perform the z-update. Thus, we are able to derive novel computational guarantees for AGBM. This is the first GBM type of algorithm with theoretically-justified accelerated convergence rate. Finally we demonstrate with a number of numerical experiments the effectiveness of AGBM over conventional GBM in obtaining a model with good training and/or testing data fidelity.

1 Introduction

Gradient Boosting Machine (GBM) [14] is a powerful supervised learning algorithm that combines multiple weak-learners into an ensemble with excellent predictive performance. GBM works very well for a number of tasks like spam filtering, online advertising, fraud detection, anomaly detection, computational physics (e.g., the Higgs Boson discovery), etc; and has routinely featured as a top algorithm in Kaggle competitions and the KDDCup [7]. GBM can naturally handle heterogeneous datasets (highly correlated data, missing data, categorical data, etc). It is also quite easy to use with several publicly available implementations: scikit-learn [29], R gbm [31], LightGBM [18], XGBoost [7], TF Boosted Trees [30], etc.

In spite of the practical success of GBM, there is a considerable gap in its theoretical understanding. The traditional interpretation of GBM is to view it as a form of steepest descent in functional space [23, 14]. While this interpretation serves as a good starting point, such framework lacks rigorous non-asymptotic convergence guarantees, especially when compared to the growing body of literature on first order convex optimization.

In convex optimization literature, Nesterov's acceleration is a successful technique to speed up the convergence of first-order methods. In this work, we show how to incorporate Nesterov momentum into the gradient boosting framework in order to obtain an accelerated gradient boosting machine.

*MIT Department of Mathematics and Operations Research Center, and Google Research. The first two authors are equal contribution.

†EPFL Computer Science Department. The first two authors are equal contribution.

‡Google Research.

§Google Research.

1.1 Our contributions

We propose the first accelerated gradient boosting algorithm that comes **with strong theoretical guarantees** and can be used with **any type of weak learner**. In particular:

- We propose a *novel* accelerated gradient boosting algorithm (AGBM) (Section 3) and prove (Section 4) that it reduces the empirical loss at a rate of $O(1/m^2)$ after m iterations, improving upon the $O(1/m)$ rate obtained by traditional gradient boosting methods.
- We propose a variant of AGBM, taking advantage of strong convexity of loss function, which achieves linear convergence (Section 5). We also list the conditions (on the loss function) under which AGBMs would be beneficial.
- With a number of numerical experiments with weak tree learners (one of the most popular type of GBMs) we confirm the effectiveness of AGBM.

Apart from theoretical contributions, we paved the way for speeding up some practical applications of GBMs, which currently require a large number of boosting iterations. For example, GBMs with boosted trees for multi-class problems are commonly implemented as a number of one-vs-rest learners, resulting in more complicated boundaries [12] and a potentially a larger number of boosting iterations required. Additionally, it is a common practice to build many very weak learners for problems where it is easy to overfit. Such large ensembles result not only in slow training time, but also slower inference. AGBMs can be potentially beneficial for all these applications.

1.2 Related Literature

Convergence Guarantees for GBM: After being first introduced by Friedman et al. [14], several works established its guaranteed convergence, without explicitly stating the convergence rate [8, 23]. Subsequently, when the loss function is both smooth and strongly convex, [3] proved an exponential convergence rate—more precisely that $O(\exp(1/\varepsilon^2))$ iterations are sufficient to ensure that the training loss is within ε of its optimal value. [35] then studied the primal-dual structure of GBM and demonstrated that in fact only $O(\log(1/\varepsilon))$ iterations are needed. However the constants in their rate were non-standard and less intuitive. This result was recently improved upon by [11] and [22], who showed a similar convergence rate but with more transparent constants such as the smoothness and strong convexity constant of the loss function, as well as the density of weak learners. Additionally, if the loss function is assumed to be smooth and convex (but not necessarily strongly convex), [22] also showed that $O(1/\varepsilon)$ iterations are sufficient. We refer the reader to [35], [11] and [22] for a more detailed literature review of the theoretical results of GBM convergence.

Accelerated Gradient Methods: For optimizing a smooth convex function, [25] showed that the standard gradient descent algorithm can be made much faster, resulting in the *accelerated* gradient descent method. While gradient descent requires $O(1/\varepsilon)$ iterations, accelerated gradient methods only require $O(1/\sqrt{\varepsilon})$. In general, this rate of convergence is optimal and cannot be improved upon [26]. Since its introduction in 1983, the mainstream research community's interest in Nesterov's accelerated method started around 15 years ago; yet even today most researchers struggle to find basic intuition as to what is really going on in accelerated methods. Such lack of intuition about the estimation sequence proof technique used by [26] has motivated many recent works trying to explain this acceleration phenomenon [33, 37, 16, 19, 15, 1, 5]. Some have recently attempted to give a physical explanation of acceleration techniques by studying the continuous-time interpretation of accelerated gradient descent via dynamical systems [33, 37, 16].

Accelerated Greedy Coordinate and Matching Pursuit Methods: Recently, [20] and [21] discussed how to accelerate matching pursuit and greedy coordinate descent algorithms respectively. Their methods however require a random step and are hence only 'semi-greedy', which does not fit in the boosting framework.

Accelerated GBM: Recently, [2] and [10] proposed accelerated versions of GBM by directly incorporating Nesterov's momentum in GBM, however, no theoretical justification was provided. Furthermore, as we argue in Section 5.2, their proposed algorithm **may not converge** to the optimum.

2 Gradient Boosting Machine

We consider a supervised learning problem with n training examples $(x_i, y_i), i = 1, \ldots, n$ such that $x_i \in \mathbb{R}^p$ is the feature vector of the i-th example and y_i is a label (in a classification problem) or a continuous response (in a regression problem). In the classical version of GBM [14], we assume we are given a base class of *learners* $\mathcal{B}$ and that our target function class is the linear combination of such base learners (denoted by $\text{lin}(\mathcal{B})$). Let $\mathcal{B} = \{b_\tau(x) \in \mathbb{R}\}$ be a family of learners parameterized by $\tau \in \mathcal{T}$. The prediction corresponding to a feature vector x is given by an additive model of the form:

$$f(x) := \left(\sum_{m=1}^{M} \beta_m b_{\tau_m}(x) \right) \in \text{lin}(\mathcal{B}) \ , \tag{1}$$

where $b_{\tau_m}(x) \in \mathcal{B}$ is a weak-learner and β_m is its corresponding additive coefficient. Here, β_m and τ_m are chosen in an adaptive fashion in order to improve the data-fidelity as discussed below. Examples of learners commonly used in practice include wavelet functions, support vector machines, and classification and regression trees [13]. We assume the set of weak learners $\mathcal{B}$ is *scalable*, namely that the following assumption holds.

Assumption 2.1. *If $b(\cdot) \in \mathcal{B}$, then $\lambda b(\cdot) \in \mathcal{B}$ for any $\lambda > 0$.*

Assumption 2.1 holds for most of the set of weak learners we are interested in. Indeed scaling a weak learner is equivalent to modifying the coefficient of the weak learner, so it does not change the structure of $\mathcal{B}$.

The goal of GBM is to obtain a good estimate of the function f that approximately minimizes the empirical loss:

$$L^\star = \min_{f \in \text{lin}(\mathcal{B})} \left\{ L(f) := \sum_{i=1}^{n} \ell(y_i, f(x_i)) \right\} \tag{2}$$

where $\ell(y_i, f(x_i))$ is a measure of the data-fidelity for the i-th sample for the loss function ℓ.

2.1 Best Fit Weak Learners

The original version of GBM by [14], presented in Algorithm 1, can be viewed as minimizing the loss function by applying an approximated steepest descent algorithm (2). GBM starts from a null function $f^0 \equiv 0$ and at each iteration computes the pseudo-residual r^m (namely, the negative gradient of the loss function with respect to the predictions so far f^m):

$$r_i^m = -\frac{d\ \ell(y_i, f^m(x_i))}{df^m(x_i)} \, . \tag{3}$$

Parameter	Dimension	Explanation
(x_i, y_i)	$\mathbb{R}^p \times \mathbb{R}$	The features and the label of the i-th sample.
X	$\mathbb{R}^{p\times n}$	$X = [x_1, x_2, \cdots, x_n]$ is the feature matrix for all training data.
$b_\tau(x)$	function	Weak learner parameterized by τ.
$b_\tau(X)$	$\mathbb{R}^n$	A vector of predictions $[b_\tau(x_i)]_i$.
$f^m(x)$	function	Ensemble of weak learners at the m-th iteration.
$f(X)$	$\mathbb{R}^n$	A vector of $[f(x_i)]_i$ for any function $f(x)$.
$g^m(x), h^m(x)$	functions	Auxiliary ensembles of weak learners at the m-th iteration.
r^m	$\mathbb{R}^n$	Pseudo residual at the m-th iteration.
c^m	$\mathbb{R}^n$	Corrected pseudo-residual at the m-th iteration.

Table 1: List of notations used.

Then a weak-learner that best fits the current pseudo-residual in terms of the least squares loss is computed as follows:

$$\tau_m = \arg\min_{\tau\in\mathcal{T}} \sum_{i=1}^{n} (r_i^m - b_\tau(x_i))^2. \tag{4}$$

This weak-learner is added to the model with a coefficient found via a line search. As the iterations progress, GBM leads to a sequence of functions $\{f^m\}_{m\in[M]}$ (where $[M]$ is a shorthand for the set $\{1, \ldots, M\}$). The usual intention of GBM is to stop early—before one is close to a minimum of Problem (2)—with the hope that such a model will lead to good predictive performance [14, 11, 38, 6].

Algorithm 1 Gradient Boosting Machine (GBM) [14]

Initialization. Initialize with $f^0(x) = 0$.
For $m = 0, \ldots, M-1$ do:
Perform Updates:
(1) Compute pseudo residual: $r^m = -\left[\frac{\partial \ell(y_i, f^m(x_i))}{\partial f^m(x_i)}\right]_{i=1,\ldots,n}$.
(2) Find the parameters of the best weak-learner: $\tau_m = \arg\min_{\tau\in\mathcal{T}} \sum_{i=1}^n (r_i^m - b_\tau(x_i))^2$.
(3) Choose the step-size η_m by line-search: $\eta_m = \arg\min_\eta \sum_{i=1}^n \ell(y_i, f^m(x_i) + \eta b_{\tau_m}(x_i))$.
(4) Update the model $f^{m+1}(x) = f^m(x) + \eta_m b_{\tau_m}(x)$.

Output. $f^M(x)$.

Perhaps the most popular set of learners are classification and regression trees (CART) [4], resulting in Gradient Boosted Decision Tree models (GBDTs). These are the models that we are using for our numerical experiments. At the same time, we would like to highlight that our algorithm is not tied to a particular type of a weak learner and is a general algorithm.

3 Accelerated Gradient Boosting Machine (AGBM)

Given the success of accelerated gradient descent as a first order optimization method, it seems natural to attempt to accelerate the GBMs. As a warm-up, we first look at how to obtain an accelerated boosting algorithm when our class of learners $\mathcal{B}$ is strong (complete) and can exactly fit any pseudo-residuals. This assumption is quite unreasonable but will serve to understand the connection between boosting and first order optimization. We then describe our actual algorithm which works for any class of *weak* learners.

3.1 Boosting with strong learners

In this subsection, we assume the class of learners $\mathcal{B}$ is *strong*, i.e. for any pseudo-residual $r \in \mathbb{R}^n$, there exists a learner $b(x) \in \mathcal{B}$ such that

$$b(x_i) = r_i, \forall\, i \in [n]\,.$$

Of course the entire point of boosting is that the learners are *weak* and thus the class is not strong, therefore this is not a realistic assumption. Nevertheless this section will provide the intuitions on how to develop AGBM.

In the GBM we compute the psuedo-residual r^m in (3) to be the negative gradient of the loss function over the predictions so far. A gradient descent step in a functional space would try to find f^{m+1} such that for $i \in \{1, \dots, n\}$

$$f^{m+1}(x_i) = f^m(x_i) + \eta r_i^m\,.$$

Here η is the step-size of our algorithm. Since our class of learners is rich, we can choose $b^m(x) \in \mathcal{B}$ to exactly satisfy the above equation.

Thus GBM (Algorithm 1) then has the following update:

$$f^{m+1} = f^m + \eta b^m\,,$$

where $b^m(x_i) = r_i^m$. In other words, GBM performs exactly functional gradient descent when the class of learners is strong, and so it converges at a rate of $O(1/m)$. Akin to the above argument, we can perform functional *accelerated* gradient descent, which has the accelerated rate of $O(1/m^2)$. In the accelerated method, we maintain three model ensembles: f, g, and h of which $f(x)$ is the only model which is finally used to make predictions during the inference time. Ensemble $h(x)$ is the *momentum* sequence and $g(x)$ is a weighted average of f and h (refer to Table 1 for list of all notations used). These sequences are updated as follows for a step-size η and $\{\theta_m = 2/(m+2)\}$:

$$\begin{aligned} g^m &= (1-\theta_m) f^m + \theta_m h^m \\ f^{m+1} &= g^m + \eta b^m \qquad &&: \text{primary model} \\ h^{m+1} &= h^m + \eta/\theta_m b^m \quad &&: \text{momentum model} \end{aligned} \tag{5}$$

where $b^m(x)$ satisfies for $i \in 1, \dots, n$

$$b^m(x_i) = -\frac{d\ \ell(y_i, g^m(x_i))}{dg^m(x_i)}\,. \tag{6}$$

Note that the psuedo-residual is computed w.r.t. g instead of f. The update above can be rewritten as

$$f^{m+1} = f^m + \eta b^m + \theta_m (h^m - f^m)\,.$$

If $\theta_m = 0$, we see that we recover the standard functional gradient descent with step-sze η. For $\theta_m \in (0, 1]$, there is an additional *momentum* in the direction of $(h^m - f^m)$.

The three sequences f, g, and h match exactly those used in typical accelerated gradient descent methods (see [26, 36] for details).

3.2 Boosting with weak learners

In this subsection, we consider the general case without assuming that the class of learners is strong. Indeed, the class of learners $\mathcal{B}$ is usually quite simple and it is very likely that for any $\tau \in \mathcal{T}$, it is

Algorithm 2 Accelerated Gradient Boosting Machine (AGBM)

Input. Starting function $f^0(x) = 0$, step-size η, momentum-parameter $\gamma \in (0, 1]$, and data $X, y = (x_i, y_i)_{i \in [n]}$.
Initialization. $h^0(x) = f^0(x)$ and sequence $\theta_m = \frac{2}{m+2}$.
For $m = 0, \dots, M-1$ do:
Perform Updates:
(1) Compute a linear combination of f and h: $g^m(x) = (1-\theta_m) f^m(x) + \theta_m h^m(x)$.
(2) Compute pseudo residual: $r^m = -\left[\frac{\partial \ell(y_i, g^m(x_i))}{\partial g^m(x_i)}\right]_{i=1,\dots,n}$.
(3) Find the best weak-learner for pseudo residual: $\tau_{m,1} = \arg\min_{\tau \in \mathcal{T}} \sum_{i=1}^n (r_i^m - b_\tau(x_i))^2$.
(4) Update the model: $f^{m+1}(x) = g^m(x) + \eta b_{\tau_{m,1}}(x)$.
(5) Update the corrected residual: $c_i^m = \begin{cases} r_i^m & \text{if } m = 0 \\ r_i^m + \frac{m+1}{m+2}(c_i^{m-1} - b_{\tau_{m-1,2}}(x_i)) & \text{o.w.} \end{cases}$.
(6) Find the best weak-learner for the corrected residual: $\tau_{m,2} = \arg\min_{\tau \in \mathcal{T}} \sum_{i=1}^n (c_i^m - b_\tau(x_i))^2$.
(7) Update the momentum model: $h^{m+1}(x) = h^m(x) + \gamma\eta/\theta_m b_{\tau_{m,2}}(x)$.

Output. $f^M(x)$.

impossible to exactly fit the residual r^m. We call this case boosting with *weak* learners. Our task then is to modify (5) to obtain a truly accelerated gradient boosting machine.

The full details are summarized in Algorithm 2 but we will highlight two key differences from (5).

First, the update to the f sequence is replaced with a weak-learner which best approximates r^m similar to (5). In particular, we compute pseudo-residual r^m computed w.r.t. g as in (6) and find a parameter $\tau_{m,1}$ such that

$$\tau_{m,1} = \arg\min_{\tau \in \mathcal{T}} \sum_{i=1}^n (r_i^m - b_\tau(x_i))^2 .$$

Secondly, and more crucially, the update to the momentum model h is decoupled from the update to the f sequence. We use an *error-corrected* pseudo-residual c^m instead of directly using r^m. Suppose that at iteration $m-1$, a weak-learner $b_{\tau_{m-1,2}}$ was added to h^{m-1}. Then error corrected residual is defined inductively as follows: for $i \in \{1, \dots, n\}$

$$c_i^m = r_i^m + \frac{m+1}{m+2}\left(c_i^{m-1} - b_{\tau_{m-1,2}}(x_i)\right) ,$$

and then we compute

$$\tau_{m,2} = \arg\min_{\tau \in \mathcal{T}} \sum_{i=1}^n (c_i^m - b_\tau(x_i))^2 .$$

Thus at each iteration *two* weak learners are computed—$b_{\tau_{m,1}}(x)$ approximates the residual r^m and the $b_{\tau_{m,2}}(x)$, which approximates the error-corrected residual c^m. Note that if our class of learners is complete then $c_i^{m-1} = b_{\tau_{m-1,2}}(x_i)$, $c^m = r^m$ and $\tau_{m,1} = \tau_{m,2}$. This would revert back to our accelerated gradient boosting algorithm for strong-learners described in (5).

4 Convergence Analysis of AGBM

We first formally define the assumptions required and then outline the computational guarantees for AGBM.

4.1 Assumptions

Let's introduce some standard regularity/continuity constraints on the loss function that we require in our analysis.

Definition 4.1. *We denote $\frac{\partial \ell(y,f)}{\partial f}$ as the derivative of the bivariant loss function ℓ w.r.t. the prediction f. We say that ℓ is σ-smooth if for any y and predictions f_1 and f_2, it holds that*

$$\ell(y, f_1) \le \ell(y, f_2) + \frac{\partial \ell(y, f_2)}{\partial f}(f_1 - f_2) + \frac{\sigma}{2}(f_1 - f_2)^2.$$

We say ℓ is μ-strongly convex (with $\mu > 0$) if for any y and predictions f_1 and f_2, it holds that

$$\ell(y, f_1) \ge \ell(y, f_2) + \frac{\partial \ell(y, f_2)}{\partial f}(f_1 - f_2) + \frac{\mu}{2}(f_1 - f_2)^2.$$

Note that $\mu \le \sigma$ always. Smoothness and strong-convexity mean that the function $l(x)$ is (respectively) upper and lower bounded by quadratic functions. Intuitively, smoothness implies that that gradient does not change abruptly and hence $l(x)$ is never 'sharp'. Strong-convexity implies that $l(x)$ always has some 'curvature' and is never 'flat'.

The notion of Minimal Cosine Angle (MCA) introduced in [22] plays a central rule in our convergence rate analysis of GBM. MCA measures how well the weak-learner $b_{\tau(r)}(X)$ approximates the desired residual r

Definition 4.2. *Let $r \in \mathbb{R}^n$ be a vector. The Minimal Cosine Angle (MCA) is defined as the similarity between r and the output of the best-fit learner $b_\tau(X)$:*

$$\Theta := \min_{r \in \mathbb{R}^n} \max_{\tau \in \mathcal{T}_m} \cos(r, b_\tau(X)) \,, \tag{7}$$

where $b_\tau(X) \in \mathbb{R}^n$ is a vector of predictions $[b_\tau(x_i)]_i$.

The quantity $\Theta \in (0, 1]$ measures how "dense" the learners are in the prediction space. For *strong* learners (in Section 3.1), the prediction space is complete, and $\Theta = 1$. For a complex space of learners $\mathcal{T}$ such as deep trees, we expect the prediction space to be dense and that $\Theta \approx 1$. For a simpler class such as tree-stumps Θ would be much smaller. Refer to [22] for a discussion of Θ.

4.2 Computational Guarantees

We are now ready to state the main theoretical result of our paper.

Theorem 4.1. *Consider Accelerated Gradient Boosting Machine (Algorithm 2). Suppose ℓ is σ-smooth, the step-size $\eta \le \frac{1}{\sigma}$ and the momentum parameter $\gamma \le \Theta^4/(4 + \Theta^2)$, where Θ is the MCA introduced in Definition 4.2. Then for all $M \ge 0$, we have:*

$$L(f^M) - L(f^*) \le \frac{1}{2\eta\gamma(M+1)^2} \|f^*(X)\|_2^2 \,.$$

Proof Sketch. Here we only give an outline—the full proof can be found in the Appendix (Section B). We use the potential-function based analysis of accelerated method (cf. [36, 37]). Recall that $\theta_m = \frac{2}{m+2}$. For the proof, we introduce the following vector sequence of auxiliary ensembles $\hat{h}$ as follows:

$$\hat{h}^0(X) = 0, \quad \hat{h}^{m+1}(X) = \hat{h}^m(X) + \frac{\eta\gamma}{\theta_m} r^m \,.$$

The sequence $\hat{h}^m(X)$ is in fact closely tied to the sequence $h^m(X)$ as we demonstrate in the Appendix (Lemma B.2). Let $f^\star$ be any function which obtains the optimal loss (2)

$$f^\star \in \underset{f \in \text{lin}(\mathcal{B})}{\arg\min} \left\{ L(f) := \sum_{i=1}^{n} \ell(y_i, f(x_i)) \right\} .$$

Let us define the following sequence of potentials:

$$V^m(f^\star) = \begin{cases} \frac{1}{2}\left\| f^\star(X) - \hat{h}^0(X) \right\|^2 & \text{if } m = 0\,, \\ \frac{\eta\gamma}{\theta_{m-1}^2}\left(L(f^m) - L^*\right) + \frac{1}{2}\left\| f^\star(X) - \hat{h}^m(X) \right\|^2 & \text{o.w} \end{cases}$$

Typical proofs of accelerated algorithms show that the potential $V^m(f^\star)$ is a *decreasing* sequence. In boosting, we use the weak-learner that fits the pseudo-residual of the loss. This can guarantee sufficient decay to the first term of $V^m(f^\star)$ related to the loss $L(f)$. However, there is no such guarantee that the same weak-learner can also provide sufficient decay to the second term as we do not apriori know the optimal ensemble $f^\star$. That is the major challenge in the development of AGBM.

We instead show that the potential decreases at least by δ_m:

$$V^{m+1}(f^\star) \le V^m(f^\star) + \delta_m \,,$$

where δ_m is an error term depending on Θ (see Lemma B.4 for the exact definition of δ_m and proof of the claim). By telescope, it holds that

$$\begin{aligned} \frac{\eta\gamma}{\theta_m^2}\left(L(f^{m+1}) - L(f^\star)\right) &\le V^{m+1}(f^*) \\ &\le \sum_{j=0}^{m} \delta_j + \frac{1}{2}\left\| f^\star(X) - \hat{h}^0(X) \right\|^2 . \end{aligned}$$

Finally a careful analysis of the error term (Lemma B.6) shows that $\sum_{j=0}^{m} \delta_j \le 0$ for any $m \ge 0$. Therefore,

$$L(f^{m+1}) - L(f^\star) \le \frac{\theta_m^2}{2\eta\gamma}\|f^\star(X)\|^2 \,,$$

which furnishes the proof by letting $m = M - 1$ and substituting the value of θ_m. □

Remark 4.1. *Theorem 4.1 implies that to get a function f^M such that the error $L(f^M) - L(f^\star) \le \varepsilon$, we need number of iterations $M \sim O\left(\frac{1}{\Theta^2\sqrt{\varepsilon}}\right)$. In contrast, standard gradient boosting machines, as proved in [22], require $M \sim O\left(\frac{1}{\Theta^2\varepsilon}\right)$ This means that for small values of ε, AGBM (Algorithm 2) can require far fewer weak learners than GBM (Algorithm 1).*

5 Extensions and Variants

In this section we study two more practical variants of AGBM. First we see how to restart the algorithm to take advantage of strong convexity of the loss function. Then we will study a straight-forward approach to accelerated GBM, which we call vanilla accelerated gradient boosting machine (VAGBM), a variant of the recently proposed algorithm in [2], however without any theoretical guarantees.

5.1 Restart and Linear Convergence

It is more common to show a linear rate of convergence for GBM methods by additionally assuming that the function $l(x)$ is μ-strongly convex (e.g. [22]). It is then relatively straight-forward to recover an *accelerated* linear rate of convergence by restarting Algorithm 2.

Algorithm 3 Accelerated Gradient Boosting Machine with Restart (AGBMR)

Input: Starting function $\tilde{f}^0(x)$, step-size η, momentum-parameter $\gamma \in (0,1]$, strong-convexity parameter μ.
For $p = 0, \ldots, P-1$ do:
(1) Run AGBM (Algorithm 2) initialized with $f^0(x) = \tilde{f}^p(x)$:
Option 1: for $M = \sqrt{\frac{2}{\eta\gamma\mu}}$ iterations.
Option 2: until $L(f^m) > L(f^{m-1})$.
(2) Set $\tilde{f}^{p+1}(x) = f^M(x)$.
Output: $\tilde{f}^P(x)$.

Theorem 5.1. *Consider Accelerated Gradient Boosting with Restarts with Option 1 (Algorithm 3) . Suppose that $l(x)$ is σ-smooth and μ-strongly convex. If the step-size $\eta \le \frac{1}{\sigma}$ and the momentum parameter $\gamma \le \Theta^4/(4+\Theta^2)$, then for any p and optimal loss $L(f^\star)$,*

$$L(\tilde{f}^{p+1}) - L^\star \le \frac{1}{2}(L(\tilde{f}^p) - L(f^\star))\,.$$

Proof. The loss function $l(x)$ is μ-strongly convex, which implies that

$$\frac{\mu}{2}\|f(X) - f^*(X)\|_2^2 \le L(f) - L(f^\star)\,.$$

Substituting this in Theorem 4.1 gives us that

$$L(f^M) - L(f^\star) \le \frac{1}{\mu\eta\gamma(M+1)^2}(L(f^0) - L(f^\star))\,.$$

Recalling that $f^0(x) = \tilde{f}^p(x)$, $f^M(x) = \tilde{f}^{p+1}(x)$, and $M^2 = 2/\eta\mu\gamma$ gives us the required statement. $\square$

The restart strategy in *Option 1* requires knowledge of the strong-convexity constant μ. Alternatively, one can also use adaptive restart strategy (*Option 2*) which is known to have good empirical performance [28].

Remark 5.1. *Theorem 5.1 shows that $M = O\left(\frac{1}{\Theta^2}\sqrt{\frac{\sigma}{\mu}}\log(1/\varepsilon)\right)$ weak learners are sufficient to obtain an error of ε using ABGMR (Algorithm 3). In contrast, standard GBM (Algorithm 1) requires $M = O\left(\frac{1}{\Theta^2}\frac{\sigma}{\mu}\log(1/\varepsilon)\right)$ weak learners. Thus AGBMR is significantly better than GBM only if the condition number is large i.e. ($\sigma/\mu \ge 1$). When $l(y,f)$ is the least-squares loss, ($\mu = \sigma = 1$) we would see no advantage of acceleration. However for more complicated functions with ($\sigma \gg \mu$) (e.g. logistic loss or exp loss), AGBMR might result in an ensemble that is significantly better (e.g. obtaining lower training loss) than that of GBM for the same number of weak learners.*

5.2 A Vanilla Accelerated Gradient Boosting Method

A natural question to ask is whether, instead of adding *two* learners at each iteration, we can get away with adding only *one*? Below we show how such an algorithm would look like and argue that it may not always converge.

Following the updates in Equation (5), we can get a direct acceleration of GBM by using the weak learner fitting the gradient. This leads to an Algorithm 4.

Algorithm 4 Vanilla Accelerated Gradient Boosting Machine (VAGBM)

Input. Starting function $f^0(x) = 0$, step-size η, momentum parameter $\gamma \in (0, 1]$.
Initialization. $h^0(x) = f^0(x)$, and sequence $\theta_m = \frac{2}{m+2}$.
For $m = 0, \ldots, M-1$ do:
Perform Updates:
(1) Compute a linear combination of f and h: $g^m(x) = (1-\theta_m)f^m(x) + \theta_m h^m(x)$.
(2) Compute pseudo residual: $r^m = -\left[\frac{\partial \ell(y_i, g^m(x_i))}{\partial g^m(x_i)}\right]_{i=1,\ldots,n}$.
(3) Find the best weak-learner for pseudo residual: $\tau_m = \arg\min_{\tau \in \mathcal{T}_m} \sum_{i=1}^n (r_i^m - b_\tau(x_i))^2$.
(4) Update the model: $f^{m+1}(x) = g^m(x) + \eta b_{\tau_m}(x)$.
(5) Update the momentum model: $h^{m+1}(x) = h^m(x) + \eta/\theta_m b_{\tau_m}(x)$.

Output. $f^M(x)$.

Algorithm 4 is equivalent to the recently developed accelerated gradient boosting machines algorithm [2, 10]. Unfortunately, it **may not always converge** to an optimum or may even **diverge**. This is because b_{τ_m} from Step (2) is only an approximate-fit to r^m, meaning that we only take an *approximate* gradient descent step. While this is not an issue in the non-accelerated version, in Step (2) of Algorithm 4, the momentum term pushes the h sequence to take a large step along the approximate gradient direction. This exacerbates the effect of the approximate direction and can lead to an additive accumulation of error as shown in [9]. In Section 6.1, we see that this is not just a theoretical concern, but that Algorithm 4 also diverges in practice in some situations.

Remark 5.2. *Our* corrected *residual c^m in Algorithm 2 was crucial to the theoretical proof of converge in Theorem 4.1. One extension could be to introduce $\gamma \in (0, 1)$ in step (5) of Algorithm 4 just as in Algorithm 2.*

6 Numerical Experiments

In this section, we present the results of computational experiments and discuss the performance of AGBM with trees as weak-learners. Subsection 6.1 demonstrates that the algorithm described in Section 5.2 may diverge numerically; Subsection 6.2 shows training and testing performance for GBM and AGBM with different parameters; and Subsection 6.3 compares the performance of GBM and AGBM with best tuned parameters. The code for the numerical experiments will be also open-sourced.

Datasets: Table 2 summaries the basic statistics of the LIBSVM datasets that were used. For each dataset, we randomly choose 80% as the training and the remaining as the testing dataset.

Dataset	task	# samples	# features
a1a	classification	1605	123
w1a	classification	2477	300
diabetes	classification	768	8
german	classification	1000	24
housing	regression	506	13
eunite2001	regression	336	16

Table 2: Basic statistics of the (real) datasets used.

AGBM with CART trees: In our experiments, all algorithms use CART trees as the weak learners. For classification problems, we use logistic loss function, and for regression problems, we use least squares loss. To reduce the computational cost, for each split and each feature, we consider 100 quantiles (instead of potentially all n values). These strategies are commonly used in implementations of GBM like [7, 30].

6.1 Evidence that VAGBM May Diverge

Figure 1 shows the training loss versus the number of trees for the housing dataset with step-size $\eta = 1$ and $\eta = 0.3$ for VAGBM and for AGBM with different parameters γ. The x-axis is number of trees added to the ensemble (recall that our AGBM algorithm adds two trees to the ensemble per iteration, so the number of boosting iterations of VAGBM and AGBM is different). As we can see, when η is large, the training loss for VAGBM diverges very fast while our AGBM with proper parameter γ converges. When η gets smaller, the training loss for VAGBM may decay faster than our AGBM at the begining, but it gets stuck and never converges to the true optimal solution. Eventually the training loss of VAGBM may even diverge. On the other hand, our theory guarantees that AGBM always converges to the optimal solution.

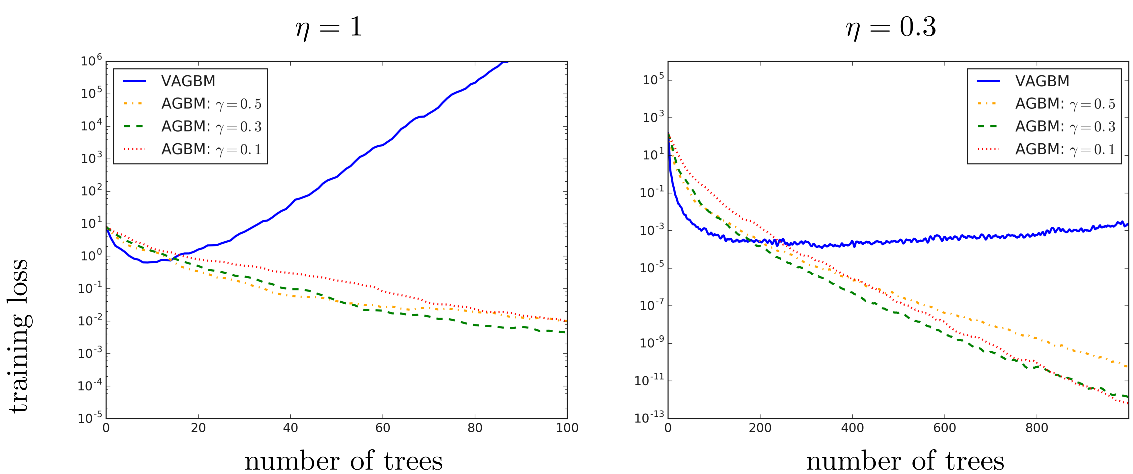

Figure 1: Training loss versus number of trees for VAGBM (which doesn't converge) and AGBM with different parameters γ.

6.2 AGBM Sensitivity to the hyperparameters

In this section we look at how the two parameters η and γ affect the performance of AGBM. Figure 2 shows the training loss and the testing loss versus the number of trees for the **a1a** dataset with two different step-sizes $\eta = 4$ and $\eta = 0.1$ (recall AGBM adds two trees per iteration). When the step-size η is large (with logistic loss, the largest step-size to guarantee the convergence is $\eta = 1/\sigma = 4$), the training loss decays very fast, and the traditional GBM can converge even faster than our AGBM at the beginning. But the testing performance is suffering, demonstrating that such a fast (due to the learning rate) convergence can result in severe overfitting. In this case, our AGBM with proper parameter γ has a slightly better testing performance. When the step-size becomes smaller, the testing performance of all algorithms becomes more stable, though the training becomes slower. AGBM with proper γ may require less number of iterations/trees to get a good training/testing performance.

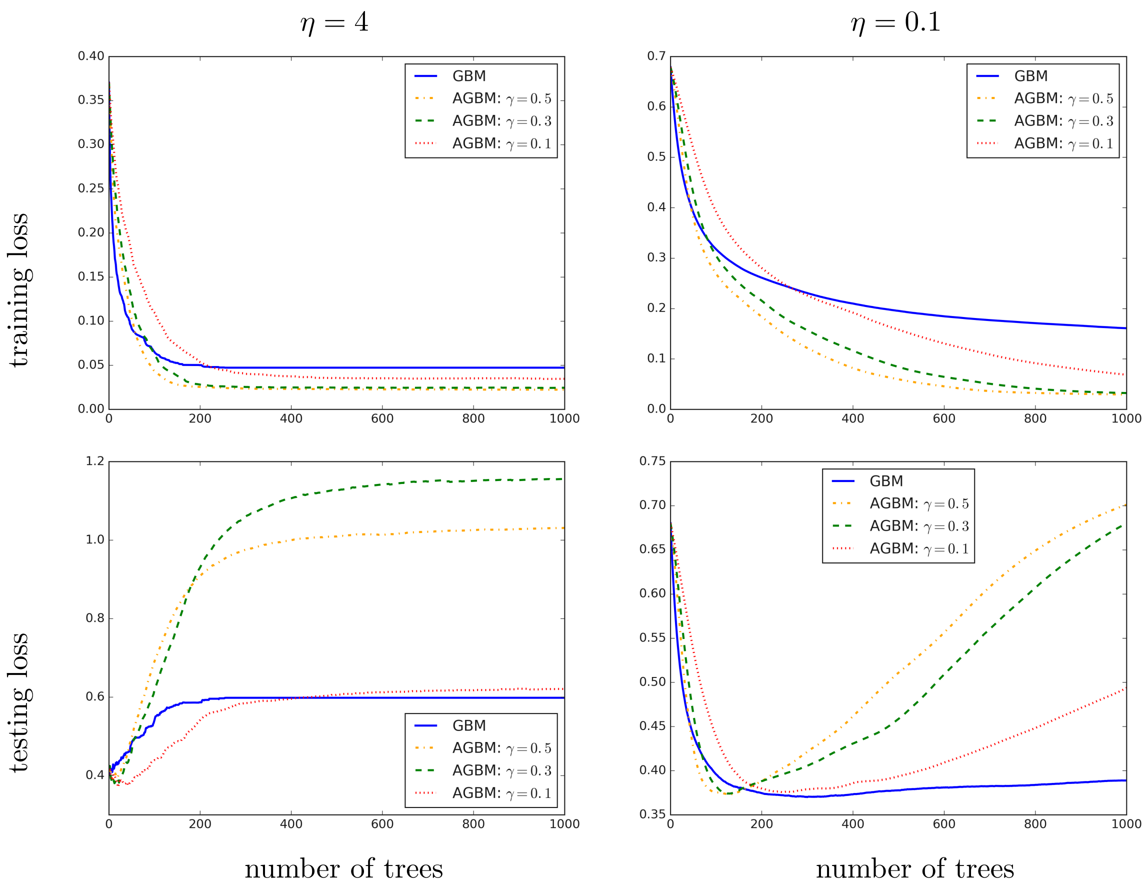

Figure 2: Training and testing loss versus number of trees for logisitc regression on a1a.

6.3 Experiments with Fine Tuning

In this section we look at the testing performance of GBM, VAGBM and AGBM on six datasets with hyperparameter tuning. We consider depth 5 trees as weak-learners. We early stop the splitting when the gain smaller than 0.001 (roughly $1/n$ for these datasets). The hyper-parameters and their ranges we tuned are:

- step size (η): $0.01, 0.03, 0.1, 0.3, 1$ for least squares loss and $0.04, 0.12, 0.4, 1.2, 4$ for logistic loss;
- number of trees: $10, 11, \ldots, 100$;
- momentum parameter γ (only for AGBM): $0.01, 0.02, 0.03, 0.05, 0.1, 0.2, 0.3, 0.5, 1$.

Dataset	Method	Training	Testing	# iter	# trees
a1a	GBM	0.2187	0.3786	97	97
	VAGBM	0.2454	**0.3661**	33	33
	AGBM	0.1994	0.3730	33	66
w1a	GBM	0.0262	0.0578	84	84
	VAGBM	0.0409	0.0578	32	32
	AGBM	0.0339	**0.0552**	47	94
diabetes	GBM	0.297	0.462	87	87
	VAGBM	0.271	0.462	24	24
	AGBM	0.297	**0.458**	47	94
german	GBM	0.244	0.505	54	54
	VAGBM	0.288	0.514	51	51
	AGBM	0.305	**0.485**	35	70
housing	GBM	0.2152	4.6603	93	93
	VAGBM	0.5676	5.8090	73	73
	AGBM	0.215	**4.5074**	35	70
eunite2001	GBM	36.73	270.1	64	64
	VAGBM	28.99	**245.2**	58	58
	AGBM	26.74	245.4	24	48

Table 3: Performance of after tuning hyper-parameters.

For each dataset, we randomly choose 80% as the training dataset and the remainder was used as the final testing dataset. We use 5-fold cross validation on the training dataset to tune the hyperparameters. Instead of going through all possible hyper-parameters, we utilize randomized search (*RandomizedSearchCV* in scikit-learn). As AGBM has more parameters (namely γ), we did proportionally more iterations of random search for AGBM. Table 3 presents the performance of GBM, VAGBM and AGBM with the tuned parameters. As we can see, the accelerated methods (AGBM and VAGBM) in general require less numbers of iterations to get similar or slightly better testing performance than GBM. Compared with VAGBM, AGBM adds two trees per iteration, and that can be more expensive, but the performance of AGBM can be more stable, for example, the testing error of VAGBM for housing dataset is much larger than AGBM.

7 Conclusion

In this paper, we proposed a novel Accelerated Gradient Boosting Machine (AGBM), prooved its rate of convergence and introduced a computationally inexpensive practical variant of AGBM that takes advantage of strong convexity of loss function and achives linear convergence. Finally we demonstrated with a number of numerical experiments the effectiveness of AGBM over conventional GBM in obtaining a model with good training and/or testing data fidelity.

A Additional Discussions

Below we include some additional discussions which could not fit into the main paper but which nevertheless help to understand the relevance of our results when applied to frameworks typically used in practice.

A.1 Line search in Boosting

Traditionally the analysis of gradient boosting methods has focused on algorithms which use line search to select the step-size η (e.g. Algorithm 1). Analysis of gradient descent suggests that is not necessary—using a fixed step-size of $1/\beta$ where $l(x)$ is β-smooth is sufficient [22]. Our accelerated Algorithm 2 also adopts this fixed step-size strategy. In fact, even the standard boosting libraries (XGBoost and TFBT) typically use a fixed (but tuned) step-size and avoid an expensive line search.

A.2 Use of Hessian

Popular boosting libraries such as XGBoost [7] and TFBT [30] compute the Hessian and perform a *Newton* boosting step instead of gradient boosting. Since the Newton step may not be well defined (e.g. if the Hessian is degenerate), an additional euclidean regularizer is also added. This has been shown to improve performance and reduce the need for a line-search for the η parameter sequence [34, 32]. For LogitBoost (i.e. when $l(x)$ is the logistic loss), [34] demonstrate that trust-region Newton's method can indeed significantly improve the convergence. Leveraging similar results in second-order methods for convex optimization (e.g. [27, 17]) and adapting accelerated second-order methods [24] would be an interesting direction for the future work.

A.3 Out-of-sample Performance

Throughout this work we focus only on minimizing the empirical training loss $L(f)$ (see Formula (2)). In reality what we really care about is the out-of-sample error of our resulting ensemble $f^M(x)$. A number of regularization tricks such as i) early stopping [38], ii) pruning [7, 30], iii) smaller step-sizes [30], iv) dropout [30] etc. are usually employed in practice to prevent over-fitting and improve generalization. Since AGBM requires much fewer iterations to achieve the same training loss than GBM, it outputs a much sparser set of learners. We believe this is partially the reason for its better out-of-sample performance. However a joint theoretical study of the out-of-sample error along with the empirical error $L_n(f)$ is much needed. It would also shed light on the effectiveness of the numerous ad-hoc regularization techniques currently employed.

B Proof of Theorem 4.1

This section proves our major theoretical result in the paper:

Theorem 4.1 Consider Accelerated Gradient Boosting Machine (Algorithm 2). Suppose ℓ is σ-smooth, the step-size $\eta \le \frac{1}{\sigma}$ and the momentum parameter $\gamma \le \Theta^4/(4+\Theta^2)$. Then for all $M \ge 0$, we have:

$$L(f^M) - L(f^*) \le \frac{1}{2\eta\gamma(M+1)^2} \|f^*(X)\|_2^2 \; .$$

$\square$

Let's start with some new notations. Define scalar constants $s = \gamma/\Theta^2$ and $t := (1-s)/2 \in (0,1)$. We mostly only need $s + t \leq 1$—the specific values of γ and t are needed only in Lemma B.6. Then define

$$\alpha_m := \frac{\eta\gamma}{\theta_m} = \frac{\eta s \Theta^2}{\theta_m},$$

then the definitions of the sequences $\{r^m\}$, $\{c^m\}$, $\hat{h}^m(X)$ and $\{\theta_m\}$ from Algorithm 3 can be simplified as:

$$\begin{aligned}
\theta_m &= \frac{2}{m+2} \\
r^m &= -\left[\frac{\partial l(y_i, g^m(x_i))}{\partial g^m(x_i)}\right]_{i=1,\dots n} \\
c^m &= r^m + (\alpha_{m-1}/\alpha_m)\left(c^{m-1} - b_{\tau^2_{m-1}}(X)\right) \\
\hat{h}^{m+1}(X) &= \hat{h}^m(X) + \alpha_m r^m .
\end{aligned}$$

The sequence $\hat{h}^m(X)$ is in fact closely tied to the sequence $h^m(X)$ as we show in the next lemma. For notational convenience, we define $c^{-1} = b_{\tau^2_{-1}}(X) = 0$ and similarly $\frac{\alpha_{-1}}{\theta_{-1}} = 0$ throughout the proof.

Lemma B.1.

$$\hat{h}^{m+1}(X) = h^{m+1}(X) + \alpha_m(c_m - b_{\tau_{m,2}}(X)) .$$

Proof. Observe that

$$\hat{h}^{m+1}(X) = \sum_{j=0}^{m} \alpha_j r^j \quad \text{and that} \quad h^{m+1}(X) = \sum_{j=0}^{m} \alpha_j b_{\tau_{j,2}}(X) .$$

Then we have

$$\begin{aligned}
\hat{h}^{m+1}(X) - h^{m+1}(X) &= \sum_{j=0}^{m} \alpha_j (r^j - b_{\tau_{j,2}}(X)) \\
&= \sum_{j=0}^{m} \alpha_j (r^j - \frac{\alpha_{j-1}}{\alpha_j} b_{\tau^2_{j-1}}(X)) - \alpha_m b_{\tau_{m,2}}(X) \\
&= \sum_{j=0}^{m} \alpha_j (c^j - \frac{\alpha_{j-1}}{\alpha_j} c^{j-1}) - \alpha_m b_{\tau_{m,2}}(X) \\
&= \sum_{j=0}^{m} (\alpha_j c^j - \alpha_{j-1} c^{j-1}) - \alpha_m b_{\tau_{m,2}}(X) \\
&= \alpha_m (c_m - b_{\tau_{m,2}}(X)) ,
\end{aligned}$$

where the third equality is due to the definition of c^m. ☐

Lemma B.2 presents the fact that there is sufficient decay of the loss function:

Lemma B.2.

$$L(f^{m+1}) \leq L(g^m) - \frac{\eta\Theta^2}{2}\|r^m\|^2 .$$

Proof. Recall that $\tau_{m,1}$ is chosen such that

$$\tau_{m,1} = \arg\min_{\tau \in \mathcal{T}} \|b_\tau(X) - r^m\|^2 .$$

Since the class of learners $\mathcal{T}$ is *scalable* (Assumption 2.1), we have

$$\begin{aligned} \|b_{\tau_{m,1}}(X) - r^m\|^2 &= \min_{\tau \in \mathcal{T}_m} \min_{\sigma \in \mathbb{R}} \|\sigma b_\tau(X) - r^m\|^2 \\ &= \|r^m\|^2 \left(1 - \arg\max_{\tau \in \mathcal{T}} cos(r^m, b_\tau(X))^2\right) \\ &\leq \|r^m\|^2 \left(1 - \Theta^2\right) , \end{aligned} \tag{8}$$

where the last inequality is because of the definition of Θ, and the second equality is due to the simple fact that for any two vectors a and b,

$$\min_{\sigma \in \mathbb{R}} \|\sigma a - b\|^2 = \|a\|^2 - \max_{\sigma \in \mathbb{R}} \left(\sigma\langle a, b\rangle - \frac{\sigma^2}{2}\|b\|^2\right) = \|a\|^2 - \|a\|^2 \frac{\langle a, b\rangle}{\|a\|^2\|b\|^2} .$$

Now recall that $L(f^{m+1}) = \sum_{i=1}^n l(y_i, f^{m+1}(x_i))$ and that $f^{m+1}(x) = g^m(x) + \eta b_{\tau_{m,1}}(x)$. Since the loss function $l(y_i, x)$ is σ-smooth and step-size $\eta \leq \frac{1}{\sigma}$, it holds that

$$\begin{aligned} L(f^{m+1}) &= \sum_{i=1}^n l(y_i, f^{m+1}(x_i)) \\ &\leq \sum_{i=1}^n l(y_i, g^m(x_i) + \eta b_{\tau_{m,1}}(x_i)) \\ &\leq \sum_{i=1}^n \left(l(y_i, g^m(x_i)) + \frac{\partial l(y_i, g^m(x_i))}{\partial g^m(x_i)} (\eta b_{\tau_{m,1}}(x_i)) + \frac{\sigma}{2} (\eta b_{\tau_{m,1}}(x_i))^2 \right) \\ &\leq \sum_{i=1}^n \left(l(y_i, g^m(x_i)) + \frac{\partial l(y_i, g^m(x_i))}{\partial g^m(x_i)} (\eta b_{\tau_{m,1}}(x_i)) + \frac{\eta}{2} (b_{\tau_{m,1}}(x_i))^2 \right) \\ &= \sum_{i=1}^n \left(l(y_i, g^m(x_i)) - r_i^m (\eta b_{\tau_{m,1}}(x_i)) + \frac{1}{2\eta} (b_{\tau_{m,1}}(x_i))^2 \right) \\ &= L(g^m) - \eta\big\langle r^m, b_{\tau_{m,1}}(X)\big\rangle + \frac{\eta}{2}\|b_{\tau_{m,1}}(X)\|^2 \\ &= L(g^m) + \frac{\eta}{2}\|b_{\tau_{m,1}}(X) - r^m\|^2 - \frac{\eta}{2}\|r^m\|^2 \\ &\leq L(g^m) - \frac{\Theta^2\eta}{2}\|r^m\|^2 , \end{aligned}$$

where the final inequality follows from (8). This furnishes the proof of the lemma. □

Lemma B.3 is a basic fact of convex function, and it is commonly used in the convergence analysis in accelerated method.

Lemma B.3. *For any function f and $m \geq 0$,*

$$L(g^m) + \theta_m\langle r^m, h^m(X) - f(X)\rangle \leq \theta_m L(f) + (1 - \theta_m)L(f^m) .$$

Proof. For any function f, it follows from the convexity of the loss function l that

$$
\begin{aligned}
L(g^m) + \langle r^m, g^m(X) - f(X)\rangle &= \sum_{i=1}^{n} l(y_i, g^m(x_i)) + \frac{\partial l(y_i, g^m(x_i))}{\partial g^m(x_i)}(f(x_i) - g^m(x_i)) \\
&\leq \sum_{i=1}^{n} l(y_i, f(x_i)) = L(f)\,. \qquad (9)
\end{aligned}
$$

Substituting $f = f^m$ in (9), we get

$$L(g^m) + \langle r^m, g^m(X) - f^m(X)\rangle \leq L(f^m)\,. \qquad (10)$$

Also recall that $g^m(X) = (1-\theta_m)f^m(X) + \theta_m h^m(X)$. This can be rewritten as

$$\theta_m(g^m(X) - h^m(X)) = (1-\theta_m)(f^m(X) - g^m(X))\,. \qquad (11)$$

Putting (9), (10), and (11) together:

$$
\begin{aligned}
&L(g^m) + \theta_m\langle r^m, h^m(X) - f(X)\rangle \\
=&L(g^m) + \theta_m\langle r^m, g^m(X) - f(X)\rangle + \theta_m\langle r^m, h^m(X) - g^m(X)\rangle \\
=&\theta_m[L(g^m) + \langle r^m, g^m(X) - f(X)\rangle] + (1-\theta_m)[L(g^m) + \langle r^m, g^m(X) - f^m(X)\rangle] \\
\leq&\theta_m L(f) + (1-\theta_m)L(f^m)\,,
\end{aligned}
$$

which furnishes the proof. □

We are ready to prove the key lemma which gives us the accelerated rate of convergence.

Lemma B.4. *Define the following potential function $V(f)$ for any given output function f:*

$$V^m(f) = \frac{\alpha_{m-1}}{\theta_{m-1}}\left(L(f^m) - L(f)\right) + \frac{1}{2}\|f(X) - \hat{h}^m(X)\|^2\,. \qquad (12)$$

At every step, the potential decreases at least by δ_m:

$$V^{m+1}(f) \leq V^m(f) + \delta_m\,,$$

where δ_m is defined as:

$$\delta_m := \frac{s\alpha_{m-1}^2}{2t}\|c^{m-1} - b_{\tau_{m-1}^2}(X)\|^2 - (1-s-t)\frac{\alpha_m^2}{2s}\|r^m\|^2\,. \qquad (13)$$

Proof. Recall that $c^{-1} = b_{\tau_{-1}^2}(X)) = 0$ and $\frac{\alpha_{-1}}{\theta_{-1}} = 0$. It follows from Lemma B.2 that:

$$
\begin{aligned}
&L(f^{m+1}) - L(g^m) + \frac{(1-s)\eta\Theta^2}{2}\|r^m\|^2 \\
\leq&-\frac{s\eta\Theta^2}{2}\|r^m\|^2 \\
=&-\alpha_m\theta_m\|r^m\|^2 + \frac{\alpha_m\theta_m}{2}\|r^m\|^2 \\
=&\theta_m\left\langle r^m, \hat{h}^m(X) - \hat{h}^{m+1}(X)\right\rangle + \frac{\theta_m}{2\alpha_m}\|\hat{h}^m(X) - \hat{h}^{m+1}(X)\|^2 \\
=&\theta_m\left\langle r^m, \hat{h}^m(X) - f(X)\right\rangle + \frac{\theta_m}{2\alpha_m}\left(\|f(X) - \hat{h}^m(X)\|^2 - \|f(X) - \hat{h}^{m+1}(X)\|^2\right)\,,
\end{aligned}
$$

where the second equality is by the definition of $\hat{h}^m(x)$ and the third is just mathematical manipulation of the equation (it is also called three-point property). By rearranging the above inequality, we have

$$
\begin{aligned}
& L(f^{m+1}) + \frac{(1-s)\eta\Theta^2}{2}\|r^m\|^2 \\
\leq & L(g^m) + \left\langle r^m, \hat{h}^m(X) - f(X) \right\rangle + \frac{\theta_m}{2\alpha_m}\left(\|f(X) - \hat{h}^m(X)\|^2 - \|f(X) - \hat{h}^{m+1}(X)\|^2 \right) \\
= & L(g^m) + \theta_m \langle r^m, h^m(X) - f(X) \rangle + \frac{\theta_m}{2\alpha_m}\left(\|f(X) - \hat{h}^m(X)\|^2 - \|f(X) - \hat{h}^{m+1}(X)\|^2 \right) + \theta_m \left\langle r^m, \hat{h}^m(X) - h^m(X) \right\rangle \\
\leq & \theta_m L(f) + (1-\theta_m) L(f^m) + \frac{\theta_m}{2\alpha_m}\left(\|f(X) - \hat{h}^m(X)\|^2 - \|f(X) - \hat{h}^{m+1}(X)\|^2 \right) + \theta_m \alpha_{m-1} \left\langle r^m, c^{m-1} - b_{\tau^2_{m-1}}(X) \right\rangle ,
\end{aligned}
$$

where the first inequality uses Lemma B.3 and the last inequality is due to the fact that $\hat{h}^m(X) - h^m(X) = \alpha_{m-1}(c^{m-1} - b_{\tau^2_{m-1}}(X))$ from Lemma B.1. Rearranging the terms and multiplying by (α_m/θ_m) leads to

$$
\begin{aligned}
& \frac{\alpha_m}{\theta_m}(L(f^{m+1}) - L(f)) + \frac{1}{2}\|f(X) - \hat{h}^{m+1}(X)\|^2 \\
\leq & \underbrace{\frac{\alpha_m(1-\theta_m)}{\theta_m}}_{:=\mathcal{A}}(L(f^m) - L(f)) + \frac{1}{2}\|f(X) - \hat{h}^m(X)\|^2 + \underbrace{\alpha_m\alpha_{m-1}\left\langle r^m, (c^{m-1} - b_{\tau^2_{m-1}}(X)) \right\rangle - \frac{(1-s)\eta\Theta^2\alpha_m}{2\theta_m}\|r^m\|^2}_{:=\mathcal{B}} .
\end{aligned}
$$

Let us examine first the term $\mathcal{A}$:

$$
\frac{\alpha_m(1-\theta_m)}{\theta_m} = (\eta\Theta^2 s)\frac{1-\theta_m}{\theta_m^2} \leq (\eta\Theta^2 s)\frac{1}{\theta_{m-1}^2} = \frac{\alpha_{m-1}}{\theta_{m-1}} .
$$

We have thus far shown that

$$
V^{m+1}(f) \leq V^m(f) + \mathcal{B} ,
$$

and we now need to show that $\mathcal{B} \leq \delta_m$. Using Mean-Value inequality, the first term in $\mathcal{B}$ can be bounded as

$$
\alpha_m\alpha_{m-1}\left\langle r^m, (c^{m-1} - b_{\tau^2_{m-1}}(X)) \right\rangle \leq \frac{\alpha_m^2 t}{2s}\|r^m\|^2 + \frac{\alpha_{m-1}^2 s}{2t}\|c^{m-1} - b_{\tau^2_{m-1}}(X)\|^2 .
$$

Substituting it in $\mathcal{B}$ shows:

$$
\begin{aligned}
\mathcal{B} &= \alpha_m\alpha_{m-1}\left\langle r^m, (c^{m-1} - b_{\tau^2_{m-1}}(X)) \right\rangle - \frac{(1-s)\eta\Theta^2\alpha_m}{2\theta_m}\|r^m\|^2 \\
&\leq \frac{\alpha_m^2 t}{2s}\|r^m\|^2 + \frac{\alpha_{m-1}^2 s}{2t}\|c^{m-1} - b_{\tau^2_{m-1}}(X)\|^2 - \frac{(1-s)\alpha_m^2}{2s}\|r^m\|^2 \\
&= \frac{\alpha_{m-1}^2 s}{2t}\|c^{m-1} - b_{\tau^2_{m-1}}(X)\|^2 - (1-s-t)\frac{\alpha_m^2}{2s}\|r^m\|^2 \\
&= \delta_m ,
\end{aligned}
$$

which finishes the proof. □

Unlike the typical proofs of accelerated algorithms, which usually shows that the potential $V^m(f)$ is a decreasing sequence, there is no guarantee that the potential $V^m(f)$ is decreasing in the boosting setting due to the use of weak learners. Instead, we are able to prove that:

Lemma B.5. *For any given m, it holds that $\sum_{j=0}^m \delta_j \leq 0$.*

Proof. We can rewrite the statement of the lemma as:

$$\sum_{j=0}^{m-1} \alpha_j^2 \|c^j - b_{\tau_{j,2}}(X)\|^2 \leq \frac{t(1-s-t)}{s^2} \sum_{j=0}^{m} \alpha_j^2 \|r^j\|^2 \,. \tag{14}$$

Here, let us focus on the term $\|c^{j+1} - b_{\tau_{j+1}^2}(X)\|^2$ for a given j. We have that

$$\begin{aligned}
\left\|c^{j+1} - b_{\tau_{j+1}^2}(X)\right\|^2 &\leq (1-\Theta^2)\left\|c^{j+1}\right\|^2 \\
&= (1-\Theta^2)\left\|r^{j+1} + \frac{\theta_{j+1}}{\theta_j}\left(c^j - b_{\tau_{j,2}}(X)\right)\right\|^2 \\
&\leq (1-\Theta^2)(1+\rho)\left\|r^{j+1}\right\|^2 + (1-\Theta^2)(1+1/\rho)\left\|\frac{\theta_{j+1}}{\theta_j}\left(c^j - b_{\tau_{j,2}}(X)\right)\right\|^2 \\
&\leq (1+\rho)(1-\Theta^2)\left\|r^{j+1}\right\|^2 + (1-\Theta^2)(1+1/\rho)\left\|\left(c^j - b_{\tau_{j,2}}(X)\right)\right\|^2 ,
\end{aligned}$$

where the first inequality follows from our assumption about the density of the weak-learner class $\mathcal{B}$ (the same of the argument in (8)), the second inequality holds for any $\rho \geq 0$ due to Mean-Value inequality, and the last inequality is from $\theta_{j+1} \leq \theta_j$. We now derives a recursive bound on the left side of (14). From this, (14) follows from an elementary fact of recursive sequence as stated in Lemma B.6 with $a_j = \alpha_j^2\left\|c^j - b_{\tau_{j,2}}(X)\right\|^2$ and $c_j = \alpha_j^2\left\|r^j\right\|^2$. □

Remark B.1. *If $c^m = b_{\tau_{m,2}}(X)$ (i.e. our class of learners $\mathcal{B}$ is* strong*), then $\delta_m = -(1-s-t)\frac{\alpha_m^2}{2s^2}\|r^m\|^2 \leq 0$.*

Lemma B.6 is an elementary fact of recursive sequence used in the proof of Lemma B.5.

Lemma B.6. *Given two sequences $\{a_j \geq 0\}$ and $\{c_j \geq 0\}$ such that the following holds for any $\rho \geq 0$,*

$$a_{j+1} \leq (1-\Theta^2)[(1+1/\rho)a_j + (1+\rho)c_{j+1}] \,,$$

then the sum of the terms a_j can be bounded as

$$\sum_{j=0}^{m} a_j \leq \frac{t(1-s-t)}{s^2} \sum_{j=0}^{m} c_j \,.$$

Proof. The recursive bound on a_j implies that

$$\begin{aligned}
a_j &\leq (1-\Theta^2)[(1+1/\rho)a_{j-1} + (1+\rho)c_j] \\
&\leq \sum_{k=0}^{j} [(1+1/\rho)(1-\Theta^2)]^{j-k}(1+\rho)(1-\Theta^2)c_k \,.
\end{aligned}$$

Summing both the terms gives

$$
\begin{aligned}
\sum_{j=0}^{m} a_j &\leq \sum_{j=0}^{m}\sum_{k=0}^{j}[(1+1/\rho)(1-\Theta^2)]^{j-k}(1+\rho)(1-\Theta^2)c_k \\
&= \sum_{k=0}^{m}\sum_{j=k}^{m}[(1+1/\rho)(1-\Theta^2)]^{j-k}(1+\rho)(1-\Theta^2)c_k \\
&\leq \sum_{k=0}^{m}\left(\sum_{j=0}^{\infty}[(1+1/\rho)(1-\Theta^2)]^{j}\right)(1+\rho)(1-\Theta^2)c_k \\
&= \frac{(1+\rho)(1-\Theta^2)}{1-(1+1/\rho)(1-\Theta^2)}\sum_{k=0}^{m} c_k \\
&= \frac{(1+\rho)(1-\Theta^2)}{\Theta^2-(1-\Theta^2)/\rho}\sum_{k=0}^{m} c_k \\
&= \frac{2(1+\rho)(1-\Theta^2)}{\Theta^2}\sum_{k=0}^{m} c_k \\
&= \frac{2(2-\Theta^2)(1-\Theta^2)}{\Theta^4}\sum_{k=0}^{m} c_k \, ,
\end{aligned}
$$

where in the last two equalities we chose $\rho = \frac{2(1-\Theta^2)}{\Theta^2}$. Now recall that $s \leq \frac{\Theta^2}{4+\Theta^2} \in (0,1)$ and that $t = (1-s)/2$:

$$
\begin{aligned}
\sum_{j=0}^{m} a_j &\leq \frac{2(2-\Theta^2)(1-\Theta^2)}{\Theta^4}\sum_{k=0}^{m} c_k \\
&\leq \frac{4}{\Theta^4}\sum_{k=0}^{m} c_k \\
&= \left(\frac{4+\Theta^2}{\Theta^2}-1\right)^2 \frac{1}{4}\sum_{k=0}^{m} c_k \\
&\leq \left(\frac{1}{s}-1\right)^2 \frac{1}{4}\sum_{k=0}^{m} c_k \\
&= \frac{(1-s)^2}{4s^2}\sum_{k=0}^{m} c_k \\
&= \frac{t(1-s-t)}{s^2}\sum_{k=0}^{m} c_k \, .
\end{aligned}
$$

□

Lemma B.4 and Lemma B.5 directly result in our major theorem:

Proof of Theorem 4.1 It follows from Lemma B.4 and Lemma B.5 that

$$V^M(f^\star) \leq V^{M-1}(f^\star) + \delta_m \leq V^0(f^\star) + \sum_{j=0}^{M-1}\delta_j \leq \frac{1}{2}\|f^0(X) - f^\star(X)\|^2 \, .$$

Notice $V^M(f^\star) \geq \frac{\alpha_{m-1}}{\theta_{m-1}}(L(f^M) - L(f^\star))$ as the term $\frac{1}{2}\|f^M(X) - f^\star(X)\|^2 \geq 0$, which induces that

$$L(f^M) - L(f^\star) \leq \frac{\theta_{M-1}}{2\alpha_{M-1}}\|f^0(X) - f^\star(X)\|^2 = \frac{1}{2\gamma\eta} \cdot \frac{\|f^0(X) - f^\star(X)\|^2}{M^2}\,.$$

□

References

[1] Zeyuan Allen-Zhu and Lorenzo Orecchia, *Linear coupling: An ultimate unification of gradient and mirror descent*, arXiv preprint arXiv:1407.1537 (2014).

[2] Gérard Biau, Benoît Cadre, and Laurent Rouvière, *Accelerated gradient boosting*, arXiv preprint arXiv:1803.02042 (2018).

[3] Peter J Bickel, Yaacov Ritov, and Alon Zakai, *Some theory for generalized boosting algorithms*, Journal of Machine Learning Research **7** (2006), no. May, 705–732.

[4] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*, Wadsworth, 1984.

[5] Sébastien Bubeck, Yin Tat Lee, and Mohit Singh, *A geometric alternative to nesterov's accelerated gradient descent*, arXiv preprint arXiv:1506.08187 (2015).

[6] Peter Bühlmann, Torsten Hothorn, et al., *Boosting algorithms: Regularization, prediction and model fitting*, Statistical Science **22** (2007), no. 4, 477–505.

[7] Tianqi Chen and Carlos Guestrin, *Xgboost: A scalable tree boosting system*, Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, ACM, 2016, pp. 785–794.

[8] Michael Collins, Robert E Schapire, and Yoram Singer, *Logistic regression, adaboost and bregman distances*, Machine Learning **48** (2002), no. 1-3, 253–285.

[9] Olivier Devolder, François Glineur, and Yurii Nesterov, *First-order methods of smooth convex optimization with inexact oracle*, Mathematical Programming **146** (2014), no. 1-2, 37–75.

[10] Erwan Fouillen, Claire Boyer, and Maxime Sangnier, *Accelerated proximal boosting*, arXiv preprint arXiv:1808.09670 (2018).

[11] Robert M Freund, Paul Grigas, Rahul Mazumder, et al., *A new perspective on boosting in linear regression via subgradient optimization and relatives*, The Annals of Statistics **45** (2017), no. 6, 2328–2364.

[12] Jerome Friedman, Trevor Hastie, and Robert Tibshirani, *Additive logistic regression: a statistical view of boosting*, Annals of Statistics **28** (1998), 2000.

[13] Jerome Friedman, Trevor Hastie, and Robert Tibshirani, *The elements of statistical learning*, vol. 1, Springer series in statistics New York, NY, USA:, 2001.

[14] Jerome H Friedman, *Greedy function approximation: a gradient boosting machine*, Annals of statistics (2001), 1189–1232.

[15] Roy Frostig, Rong Ge, Sham Kakade, and Aaron Sidford, *Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization*, International Conference on Machine Learning, 2015.

[16] Bin Hu and Laurent Lessard, *Dissipativity theory for Nesterov's accelerated method*, arXiv preprint arXiv:1706.04381 (2017).

[17] Sai Praneeth Karimireddy, Sebastian U Stich, and Martin Jaggi, *Global linear convergence of newton's method without strong-convexity or lipschitz gradients*, arXiv preprint arXiv:1806.00413 (2018).

[18] 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*, Advances in Neural Information Processing Systems, 2017, pp. 3146–3154.

[19] Hongzhou Lin, Julien Mairal, and Zaid Harchaoui, *A universal catalyst for first-order optimization*, Advances in Neural Information Processing Systems, 2015.

[20] Francesco Locatello, Anant Raj, Sai Praneeth Karimireddy, Gunnar Rätsch, Bernhard Schölkopf, Sebastian Stich, and Martin Jaggi, *On matching pursuit and coordinate descent*, International Conference on Machine Learning, 2018, pp. 3204–3213.

[21] Haihao Lu, Robert M Freund, and Vahab Mirrokni, *Accelerating greedy coordinate descent methods*, arXiv preprint arXiv:1806.02476 (2018).

[22] Haihao Lu and Rahul Mazumder, *Randomized gradient boosting machine*, arXiv preprint arXiv:1810.10158 (2018).

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

[24] Yu Nesterov, *Accelerating the cubic regularization of newtons method on convex problems*, Mathematical Programming **112** (2008), no. 1, 159–181.

[25] Yurii Nesterov, *A method of solving a convex programming problem with convergence rate* $O(1/k^2)$, Soviet Mathematics Doklady, vol. 27, 1983, pp. 372–376.

[26] ______, *Introductory lectures on convex optimization: A basic course*, vol. 87, Springer Science & Business Media, 2004.

[27] Yurii Nesterov and Boris T Polyak, *Cubic regularization of newton method and its global performance*, Mathematical Programming **108** (2006), no. 1, 177–205.

[28] Brendan Odonoghue and Emmanuel Candes, *Adaptive restart for accelerated gradient schemes*, Foundations of computational mathematics **15** (2015), no. 3, 715–732.

[29] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al., *Scikit-learn: Machine learning in python*, Journal of machine learning research **12** (2011), no. Oct, 2825–2830.

[30] Natalia Ponomareva, Soroush Radpour, Gilbert Hendry, Salem Haykal, Thomas Colthurst, Petr Mitrichev, and Alexander Grushetsky, *Tf boosted trees: A scalable tensorflow based framework for gradient boosting*, Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2017, pp. 423–427.

[31] Greg Ridgeway, Maintainer Harry Southworth, and Suggests RUnit, *Package gbm*, Viitattu **10** (2013), no. 2013, 40.

[32] Fabio Sigrist, *Gradient and newton boosting for classification and regression*, arXiv preprint arXiv:1808.03064 (2018).

[33] Weijie Su, Stephen Boyd, and Emmanuel J Candes, *A differential equation for modeling Nesterovs accelerated gradient method: theory and insights*, Journal of Machine Learning Research **17** (2016), no. 153, 1–43.

[34] Peng Sun, Tong Zhang, and Jie Zhou, *A convergence rate analysis for logitboost, mart and their variant.*, ICML, 2014, pp. 1251–1259.

[35] Matus Telgarsky, *A primal-dual convergence analysis of boosting*, Journal of Machine Learning Research **13** (2012), no. Mar, 561–606.

[36] P Tseng, *On accelerated proximal gradient methods for convex–concave optimization preprint*, 2008.

[37] Ashia C Wilson, Benjamin Recht, and Michael I Jordan, *A Lyapunov analysis of momentum methods in optimization*, arXiv preprint arXiv:1611.02635 (2016).

[38] Tong Zhang and Bin Yu, *Boosting with early stopping: Convergence and consistency*, The Annals of Statistics **33** (2005), no. 4, 1538–1579.