
Accelerating Gradient Boosting Machines

Haihao Lu*
Google,
University of Chicago

Sai Praneeth Karimireddy*
EPFL

Natalia Ponomareva
Google

Vahab Mirrokni
Google

Abstract

Gradient Boosting Machine (GBM) introduced by Friedman (2001) is a widely popular ensembling technique and is routinely used in competitions such as Kaggle and the KDD-Cup (Chen and Guestrin, 2016). In this work, we propose an 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, with naive application, the errors can accumulate in the momentum term. To overcome it, we design a “corrected pseudo residual” that serves as a new target for fitting a weak learner, 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 a theoretically-justified accelerated convergence rate.

1 Introduction

Gradient Boosting Machine (GBM) (Friedman, 2001) is an iterative ensembling procedure for supervised tasks (classification or regression) which combines multiple weak-learners to create a strong ensemble. GBM has excellent practical performance and is a staple tool used in Kaggle and the KDDCup (Chen and Guestrin, 2016). Its popularity can be attributed to its flexibility—it naturally supports heterogeneous data and tasks—and has several open-source implementations: scikit-learn (Pedregosa et al., 2011), R gbm (Ridgeway et al., 2013), LightGBM (Ke et al., 2017), XGBoost (Chen and Guestrin, 2016), TF Boosted Trees (Ponomareva

et al., 2017), etc.

Despite its popularity, the theoretical analysis of the method is unsatisfactory. GBM is typically interpreted as an iterative *functional gradient descent* (Mason et al., 2000; Friedman, 2001), but lacks rigorous finite-time convergence guarantees. In this work, we use this viewpoint as a starting point and try to apply well-studied techniques from first-order convex optimization.

In convex optimization literature, Nesterov’s acceleration is a successful technique used 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 (AGBM). This paves 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 often implemented as a number of one-vs-rest learners, resulting in more complicated boundaries Friedman et al. (1998) and a potentially a larger number of boosting iterations required. Additionally, it is common practice to build many very-weak learners (for example oblivious trees) 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 such applications.

Our main contribution is the first accelerated gradient boosting algorithm that comes **with strong theoretical guarantees** and which can be used with any type of weak learner. We introduce our algorithm in 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.

Related Literature.

GBM Convergence Guarantees: After being first introduced by Friedman (2001), several works established its guaranteed convergence, without explicitly

* Equal contribution.

stating the convergence rate (Collins et al., 2002; Mason et al., 2000). Subsequently, when the loss function is both smooth and strongly convex, Bickel et al. (2006) proved a slow 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. Telgarsky (2012) 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 Freund et al. (2017) and Lu and Mazumder (2018), 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 smooth and convex (not necessarily strongly convex), Lu and Mazumder (2018) showed that $O(1/\varepsilon)$ iterations suffice. Please refer to Telgarsky (2012), Freund et al. (2017), Lu and Mazumder (2018) for a review of theoretical results of GBM convergence.

Accelerated Gradient Methods: For optimizing a smooth convex function, Nesterov (1983) showed that the standard gradient descent (GD) algorithm can be made much faster, resulting in the *accelerated* gradient descent method. While GD requires $O(1/\varepsilon)$ iterations, accelerated gradient methods only require $O(1/\sqrt{\varepsilon})$. This rate of convergence is optimal and cannot be improved upon Nesterov (2004). The mainstream research community’s interest in Nesterov’s method started only 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 Nesterov (2004) has motivated many recent works trying to explain this acceleration phenomenon (Su et al., 2016; Wilson et al., 2016; Hu and Lessard, 2017; Lin et al., 2015; Frostig et al., 2015; Allen-Zhu and Orecchia, 2014; Bubeck et al., 2015; Chambolle and Dossal, 2015). There are also attempts to give a physical explanation of acceleration by studying the continuous-time interpretation of accelerated GD via dynamical systems (Su et al., 2016; Wilson et al., 2016; Hu and Lessard, 2017).

Accelerated Greedy Coordinate Descent and Matching Pursuit Methods: GBM can be viewed as a greedy coordinate descent algorithm or a matching pursuit algorithm in transformed spaces. Recently, Lu et al. (2018) and Locatello et al. (2018) discussed how to accelerate greedy coordinate descent and matching pursuit algorithms respectively. Their methods however require a random step and are hence only ‘semi-greedy’, which does not fit in to the boosting framework.

Accelerated GBM: Very recently, Biau et al. (2018)

and Fouillen et al. (2018) proposed accelerated versions of GBM by directly incorporating Nesterov’s momentum in GBM, but without theoretical justification. Furthermore, as we show in Section 5.1, their proposed algorithm **may not converge** to the optimum.

2 Gradient Boosting Machine

We consider a supervised learning problem with n training examples $(x_i, y_i), i = 1, \dots, n$ 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 (Friedman, 2001), 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:

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

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 (Friedman et al., 2001). 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:

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

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 (Friedman, 2001), presented in Algorithm 1, can be viewed as minimizing the loss function by applying an approximated steepest

descent algorithm to the loss in (2). GBM starts from a null function $f^0 \equiv 0$ and at each iteration m computes the pseudo-residual r^m (namely, the negative gradient of the loss function with respect to the predictions so far f^m), then a weak-learner that best fits the current pseudo-residual in terms of the least squares loss is computed.

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 $\{1, \dots, M\}$). 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 (Friedman, 2001; Freund et al., 2017; Zhang and Yu, 2005; Bühlmann et al., 2007).

Algorithm 1 Gradient Boosting Machine (GBM) (Friedman, 2001)

Initialization. Initialize with $f^0(x) = 0$.

For $m = 0, \dots, M - 1$ do:

Perform Updates:

(1) Compute pseudo residual: $r^m = - \left[\frac{\partial \ell(y_i, f^m(x_i))}{\partial f^m(x_i)} \right]_{i=1, \dots, n}$.

(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) (Breiman et al., 1984), resulting in Gradient Boosted Decision Tree models (GBDTs). We will use GBDTs 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 GD as a first order optimization method, it seems natural to attempt to accelerate the GBMs. To start, we first look at how to obtain an accelerated boosting algorithm when our class of learners \mathcal{B} is **strong** (i.e. 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 proceed with an algorithm that works for any class of **weak** learners.

3.1 First Attempt: 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 pseudo-residual r^m to be the negative gradient of the loss function over the predictions so far. A gradient descent step in a functional space would define f^{m+1} as, 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 $\{\theta_m = 2/(m+2)\}$:

$$\begin{aligned} g^m &= (1 - \theta_m) f^m + \theta_m h^m \\ f^{m+1} &= g^m + \eta b^m && : \text{primary model} \\ h^{m+1} &= h^m + \eta / \theta_m b^m && : \text{momentum model} \end{aligned} \quad (3)$$

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

$$b^m(x_i) = - \frac{\partial \ell(y_i, g^m(x_i))}{\partial g^m(x_i)}. \quad (4)$$

Note that the pseudo-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-size 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 Nesterov (2004); Tseng (2008) for details).

Parameter	Dimension	Explanation
(x_i, y_i)	$\mathbb{R}^p \times \mathbb{R}$	The features and the label of the i -th sample.
X	$\mathbb{R}^{p \times n}$	$X = [x_1, x_2, \dots, x_n]$ 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.

3.2 Main Setting: 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 impossible to exactly fit the residual r^m . We call this case boosting with **weak** learners. Our task then is to modify (3) to obtain a truly accelerated gradient boosting machine.

The full details are summarized in Algorithm 2 but we will highlight two key differences from (3). First, the update to the f sequence is replaced with a weak-learner which best approximates r^m similar to step 2 of Algorithm 1. In particular, we compute pseudo-residual r^m computed w.r.t. g as in (4) 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 $i \in \{1, \dots, n\}$

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

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 (i.e. strong), 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 (3).

The difficulty of accelerating boosting with weak learners is the error made during the fitting of the pseudo-

residual. The momentum term ends up taking large steps, thereby ‘amplifying’ and accumulating such error. Using error-correction helps to correct the error from the past steps, ensuring that the error remains controlled.

In Algorithm 2, we utilize a new hyper-parameter γ , which is called momentum-parameter throughout the paper. In practice, the performance of AGBM is not too sensitive to the momentum-parameter γ and can be manually picked to lie between $(0, 1)$.

4 Convergence Analysis of AGBM

We first list 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 that we use in our analysis.

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

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

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

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

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

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. $f^M(x)$.

The notion of Minimal Cosine Angle (MCA) introduced in Lu and Mazumder (2018) plays a central role in our convergence rate analysis of GBM. MCA measures how well the weak-learner $b_\tau(X)$ 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 $b_\tau(X)$:

$$\Theta := \min_{r \in \mathbb{R}^n} \max_{\tau \in \mathcal{T}} \cos(r, b_\tau(X)), \quad (5)$$

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.

It is also straightforward to extend the definition of Θ (and hence all our convergence results) to approximate fitting of weak learners. Such a relaxation is necessary since computing the exact best-fit weak learner is often computationally prohibitive. Refer to Lu and Mazumder (2018) for a more thorough discussion of Θ .

4.2 Computational Guarantees

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

Theorem 4.1. Consider the Accelerated Gradient Boosting Machine (Algorithm 2). Suppose ℓ is σ -smooth, the step-size $\eta \leq \frac{1}{\sigma}$ and the momentum parameter $\gamma \leq \Theta^4 / (4 + \Theta^2)$, where Θ is the MCA introduced

in Definition 4.2. Then for all $M \geq 0$, we have:

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

Proof Sketch. Here we only give an outline—the full proof can be found in the Appendix (Section C). We use the potential-function based analysis of accelerated method (cf. Tseng (2008); Wilson et al. (2016)). 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 C.2). Let f^* be any function which obtains the optimal loss (2)

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

Let us define the following sequence of potentials:

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

Typical proofs of accelerated algorithms show that the potential $V^m(f^*)$ 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 $V^m(f^*)$ 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^* . That is the major challenge in the development of AGBM.

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

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

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

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

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

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

which finishes the proof by letting $m = M - 1$ and substituting the value of θ_m . \square

Remark 4.1. *Theorem 4.1 implies that to get a function f^M such that the error $L(f^M) - L(f^*) \leq \varepsilon$, we need number of iterations $M = O\left(\frac{1}{\Theta^2\sqrt{\varepsilon}}\right)$. In contrast, standard gradient boosting machines, as proved in Lu and Mazumder (2018), require $M = O\left(\frac{1}{\Theta^2\varepsilon}\right)$. This means that for small values of ε , AGBM (Algorithm 2) can require far fewer weak learners than GBM (Algorithm 1).*

The next Theorem presents an *accelerated* linear rate of convergence by restarting Algorithm 2 for minimizing strongly convex loss function $l(x)$.

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 $p = 0, \dots, P - 1$ 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 4.2. *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 $\eta \leq \frac{1}{\sigma}$ and the momentum parameter*

$\gamma \leq \Theta^4/(4 + \Theta^2)$, then for any p and optimal loss $L(f^*)$,

$$L(\tilde{f}^{p+1}) - L^* \leq \frac{1}{2}(L(\tilde{f}^p) - L(f^*)).$$

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

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

Substituting this in Theorem 4.1 gives us that

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

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 (O’Donoghue and Candes, 2015).

Remark 4.2. *Theorem 4.2 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 AGBMR (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. ($\sigma/\mu \geq 1$). 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 Numerical Experiments

In this section, we present the results of computational experiments and discuss the performance of AGBM with trees as weak-learners. Subsection 5.1 discusses the necessity of the error-corrected residual in AGBM. Subsection 5.2 shows training and testing performance for GBM and AGBM with different parameters. Subsection 5.3 compares the performance of GBM and AGBM with best tuned parameters. The code for the numerical experiments will be released.

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

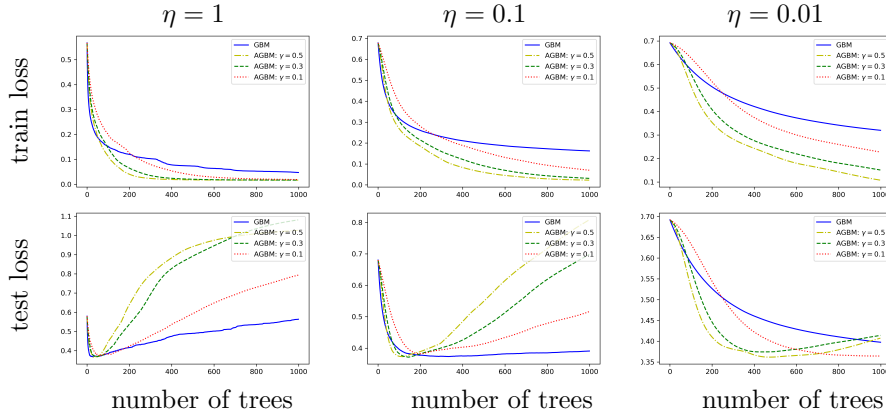


Figure 1: Training and testing loss versus number of trees for logistic regression on `a1a`.

each feature, we consider 100 quantiles (instead of potentially all n values). These strategies are commonly used in implementations of GBM like Chen and Guestrin (2016); Ponomareva et al. (2017).

5.1 Vanilla Accelerated Gradient Boosting (VAGM)

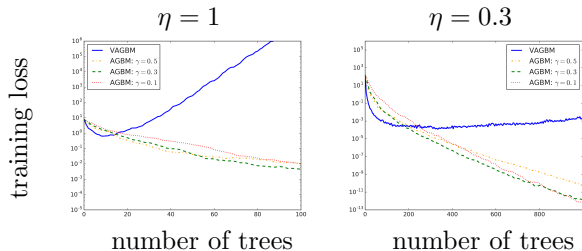


Figure 2: Training loss versus number of trees for VAGBM (which doesn’t converge) and AGBM with different parameters γ .

A more straightforward way of incorporating Nesterov momentum in boosting (which we refer to as vanilla AGBM or VAGBM) is explored in Biau et al. (2018) and Fouillen et al. (2018). VAGBM adds only one base weak-learner in each iteration as opposed to AGBM which adds two. Unfortunately, VAGBM may not always converge to the optimum as we empirically demonstrate here. A more theoretical discussion of VAGBM is presented in Section B.1.

Figure 2 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 beginning, 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.

5.2 Typical Performance of AGBM

Figure 1 presents the training loss and the testing loss of GBM and AGBM (with three γ values) versus the number of trees for the `a1a` dataset with three different learning rate $\eta = 1$, $\eta = 0.1$ and $\eta = 0.01$ (recall that AGBM adds two trees per iteration). It can be seen clearly that AGBM has faster training performance than GBM for all learning rates η , empirically showcasing the difference between convergence rates of $O(1/m^2)$ and $O(1/m)$. The training loss in general decays faster with larger learning rate η .

On test, all algorithms eventually overfit. However, AGBM can overfit in an earlier stage than GBM and seems to be more sensitive to number of trees added. This is because the training loss of AGBM decays too fast and the variance takes control in the testing loss. This seems to indicate that overfitting on test loss accompanies faster convergence on training loss. However, this issue can easily be circumvented by using early stopping—the best test loss of AGBM is comparable to that of GBM. In fact, AGBM with early stopping may require fewer iterations/trees than GBM to get similar training/testing performance.

5.3 Experiments with Fine Tuning

We evaluate AGBM and GBM on a number of small datasets, fixing the number of trees, depth and η and tuning other hyper-parameters. See Section A for details. The results are tabulated in Table 2. As we can see, the accelerated method in general is beneficial for

# trees	Dataset	AGBM		GBM	
		Training	Testing	Training	Testing
30	diabetes	0.3760+/-0.0254	0.5018 +/- 0.0335	0.5055+/-0.0084	0.5364+/-0.0287
	german	0.4076+/-0.0153	0.5308 +/- 0.0182	0.5319+/-0.0044	0.5713+/- 0.0144
	housing	2.0187+/-0.2726	7.3432+/-3.0826	2.3173+/-0.1177	4.9773 +/-2.0395
	w1a	0.1840+/-0.0013	0.1949 +/- 0.0093	0.2886+/-0.0029	0.2903+/- 0.0065
	a1a	0.3611+/-0.0090	0.4128 +/- 0.0188	0.4647+/-0.0052	0.4761+/- 0.0128
	sonar	0.1864+/-0.0108	0.4627 +/- 0.0548	0.3789+/-0.0185	0.5403+/-0.0367
50	diabetes	0.3487+/-0.0516	0.4869 +/-0.0390	0.4620 +/- 0.0060	0.5050+/-0.0348
	german	0.3695+/-0.0167	0.5114 +/- 0.0287	0.4911+/- 0.0057	0.5482+/- 0.0169
	housing	1.1388+/-0.2424	5.6229+/- 1.9212	1.4675+/- 0.1303	4.7233 +/- 2.9004
	w1a	0.0743+/-0.0015	0.1014 +/-0.0161	0.2087+/- 0.0037	0.2121+/-0.0091
	a1a	0.2812+/-0.0147	0.3686 +/- 0.0306	0.4144+/- 0.0063	0.4326+/- 0.0175
	sonar	0.0562+/-0.0053	0.3768 +/- 0.0077	0.2842+/- 0.0165	0.4981+/- 0.0257
100	diabetes	0.3119+/-0.0430	0.4937+/-0.0459	0.4130+/-0.0175	0.4797 +/-0.0409
	german	0.3569+/-0.0304	0.5175 +/-0.0248	0.4364+/- 0.0089	0.5280+/-0.0203
	housing	0.6868+/-0.2020	5.0862+/-2.0913	0.8779 +/-0.1072	4.4168 +/-2.7163
	w1a	0.0409+/-0.0034	0.0647 +/-0.0128	0.1333+/- 0.0039	0.1396+/-0.0121
	a1a	0.2797+/-0.0132	0.3675 +/-0.0363	0.3575+/- 0.0057	0.3914+/-0.0232
	sonar	0.0225+/-0.0179	0.3540 +/-0.0787	0.1902+/- 0.0637	0.4664+/-0.0660

Table 2: Performance after tuning hyper-parameters on a representative sample of data-sets.

underfitting scenarios (30 and 50 trees). Housing is a small dataset where AGBM seems to overfit quickly. For such small datasets, 100 weak learners start to overfit, and accelerated method overfits faster, as expected.

6 Additional Discussions

Below we explain relevance of our results when applied to frameworks typically used in practice.

6.1 Use of Hessian

Popular boosting libraries such as XGBoost (Chen and Guestrin, 2016) and TFBT (Ponomareva et al., 2017) 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 (Sun et al., 2014; Sigrist, 2018). For LogitBoost (i.e. when $l(x)$ is the logistic loss), Sun et al. (2014) 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. Nesterov and Polyak (2006); Karimireddy et al. (2018)) and adapting accelerated second-order methods Nesterov (2008) would be an interesting direction for the future work.

6.2 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 (Zhang

and Yu, 2005), ii) pruning (Chen and Guestrin, 2016; Ponomareva et al., 2017), iii) smaller step-sizes (Ponomareva et al., 2017), iv) dropout (Ponomareva et al., 2017) etc. are usually employed in practice to prevent overfitting 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. At the same time, it is common to slow down learning process (for example using smaller learning rate and weaker trees) to reduce overfitting on small dataset (but train for longer and have a larger ensemble). From preliminary experimental studies we see that AGBM overfits fast on small datasets and should be used with early stopping or more aggressive regularization. However, faster learning should be beneficial for large datasets and complex decision functions, where AGBM can deliver much smaller ensembles with good performance. A joint theoretical study of the out-of-sample error along with the empirical error $L_n(f)$ in the style of Zhang and Yu (2005) is much needed.

7 Conclusion

In this paper, we proposed a novel Accelerated Gradient Boosting Machine (AGBM) which can be used with any type of weak learners and which provably converges faster than the traditional Gradient Boosting Machine (GBM). We also ran preliminary experiments and demonstrated that AGBM indeed converges significantly faster than GBM on the training (empirical) loss and can match or improve upon GBM test loss. In practice, however, boosting methods are equipped with a number of additional heuristics which improve the test error. A systematic analysis of such heuristics, and incorporating them into the AGBM framework are promising directions for future work.

References

- Allen-Zhu, Z. and Orecchia, L. (2014). Linear coupling: An ultimate unification of gradient and mirror descent. *arXiv preprint arXiv:1407.1537*.
- Biau, G., Cadre, B., and Rouvière, L. (2018). Accelerated gradient boosting. *arXiv preprint arXiv:1803.02042*.
- Bickel, P. J., Ritov, Y., and Zakai, A. (2006). Some theory for generalized boosting algorithms. *Journal of Machine Learning Research*, 7(May):705–732.
- Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth.
- Bubeck, S., Lee, Y. T., and Singh, M. (2015). A geometric alternative to nesterov’s accelerated gradient descent. *arXiv preprint arXiv:1506.08187*.
- Bühlmann, P., Hothorn, T., et al. (2007). Boosting algorithms: Regularization, prediction and model fitting. *Statistical Science*, 22(4):477–505.
- Chambolle, A. and Dossal, C. (2015). On the convergence of the iterates of the “fast iterative shrinkage/thresholding algorithm”. *Journal of Optimization theory and Applications*, 166(3):968–982.
- Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM.
- Collins, M., Schapire, R. E., and Singer, Y. (2002). Logistic regression, adaboost and bregman distances. *Machine Learning*, 48(1-3):253–285.
- Devolder, O., Glineur, F., and Nesterov, Y. (2014). First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, 146(1-2):37–75.
- Fouillen, E., Boyer, C., and Sangnier, M. (2018). Accelerated proximal boosting. *arXiv preprint arXiv:1808.09670*.
- Freund, R. M., Grigas, P., Mazumder, R., et al. (2017). A new perspective on boosting in linear regression via subgradient optimization and relatives. *The Annals of Statistics*, 45(6):2328–2364.
- Friedman, J., Hastie, T., and Tibshirani, R. (1998). Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000.
- Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 10. Springer series in statistics New York, NY, USA:.
- Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.
- Frostig, R., Ge, R., Kakade, S., and Sidford, A. (2015). Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization. In *International Conference on Machine Learning*.
- Hu, B. and Lessard, L. (2017). Dissipativity theory for Nesterov’s accelerated method. *arXiv preprint arXiv:1706.04381*.
- Karimireddy, S. P., Stich, S. U., and Jaggi, M. (2018). Global linear convergence of newton’s method without strong-convexity or lipschitz gradients. *arXiv preprint arXiv:1806.00413*.
- Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154.
- Lin, H., Mairal, J., and Harchaoui, Z. (2015). A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems*.
- Locatello, F., Raj, A., Karimireddy, S. P., Rätsch, G., Schölkopf, B., Stich, S., and Jaggi, M. (2018). On matching pursuit and coordinate descent. In *International Conference on Machine Learning*, pages 3204–3213.
- Lu, H., Freund, R. M., and Mirrokni, V. (2018). Accelerating greedy coordinate descent methods. *arXiv preprint arXiv:1806.02476*.
- Lu, H. and Mazumder, R. (2018). Randomized gradient boosting machine. *arXiv preprint arXiv:1810.10158*.
- Mason, L., Baxter, J., Bartlett, P. L., and Frean, M. R. (2000). Boosting algorithms as gradient descent. In *Advances in neural information processing systems*, pages 512–518.
- Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate $O(1/k^2)$. In *Soviet Mathematics Doklady*, volume 27, pages 372–376.
- Nesterov, Y. (2004). *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media.
- Nesterov, Y. (2008). Accelerating the cubic regularization of newton’s method on convex problems. *Mathematical Programming*, 112(1):159–181.
- Nesterov, Y. and Polyak, B. T. (2006). Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205.
- O’donoghue, B. and Candes, E. (2015). Adaptive restart for accelerated gradient schemes. *Foundations of computational mathematics*, 15(3):715–732.

- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.
- Ponomareva, N., Radpour, S., Hendry, G., Haykal, S., Colthurst, T., Mitrichev, P., and Grushetsky, A. (2017). Tf boosted trees: A scalable tensorflow based framework for gradient boosting. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 423–427. Springer.
- Ridgeway, G., Southworth, M. H., and RUnit, S. (2013). Package ‘gbm’. *Viitattu*, 10(2013):40.
- Sigrist, F. (2018). Gradient and newton boosting for classification and regression. *arXiv preprint arXiv:1808.03064*.
- Su, W., Boyd, S., and Candes, E. J. (2016). A differential equation for modeling Nesterov’s accelerated gradient method: theory and insights. *Journal of Machine Learning Research*, 17(153):1–43.
- Sun, P., Zhang, T., and Zhou, J. (2014). A convergence rate analysis for logitboost, mart and their variant. In *ICML*, pages 1251–1259.
- Telgarsky, M. (2012). A primal-dual convergence analysis of boosting. *Journal of Machine Learning Research*, 13(Mar):561–606.
- Tseng, P. (2008). On accelerated proximal gradient methods for convex–concave optimization preprint.
- Wilson, A. C., Recht, B., and Jordan, M. I. (2016). A Lyapunov analysis of momentum methods in optimization. *arXiv preprint arXiv:1611.02635*.
- Zhang, T. and Yu, B. (2005). Boosting with early stopping: Convergence and consistency. *The Annals of Statistics*, 33(4):1538–1579.