Gradient boosting machines: a comparison between different approaches

Lorenzo Tausani
lorenzo.tausani@studenti.unipd.it

Francesco Maria Calistroni
francescomaria.calistroni@studenti.unipd.it

Giuseppe Capizzi
giuseppe.capizzi@studenti.unipd.it

Elisa Tremolada
elisa.tremolada@studenti.unipd.it

June 22, 2021

1 Introduction

As discussed during the lectures, many data science and machine learning applications can be viewed as Expected Risk Minimization problems: given two spaces of objects X and Y, attempt to learn a function (the so-called *hypothesis*) which outputs an object $y \in Y$ given $x \in X$. Such problems are of two main types: regression and classification; in the latter case, refer to Friedman's paper for a discussion of how the prediction function acquires the parameterized form $h(x; w)$ where w is the set of parameters.

The Expected Risk Minimization problem, from an optimization point of view, is that of finding the parameters defining the prediction function which minimizes the losses caused by inaccurate predictions, calculated by means of a loss function $l(h(x; w), y)$. However, to minimize the value of the so-called expected loss given by $\mathbf{E}_{xy}(l(h(x; w)); y)$, we would need to know the unknown distribution of x and y (denoted by xy). Therefore, in real world applications, one attempts to approximate the objective function by means of a form of sample average approximation, and the problem becomes one of Empirical Risk Minimization.

Table 1

Expected Risk Minimization	$\min_w R_w = \mathbf{E}_{xy}(l(h(x; w)); y)$
Empirical Risk Minimization	$\min_w R_w(w) = \frac{1}{m} \sum_{i=1}^{m} (l(h(x; w)); y)*$
$*i = 1, ..., m$ indicate the indexes of the (xy) samples	

1

In this deterministic framework, gradient descent methods become the obvious choice for implementing solutions to the above problems. Various methods for improving the convergence rate of the classic gradient descent algorithm have been developed over the years; here, we focus on Friedman's Gradient Boosting Machine and some of its most recent variants.

Gradient Boosting Machine (GBM) introduced by Friedman is a widely popular technique for constructing prediction machines by ensembling weak learners (Friedman, 2001). In this paper, we analyse and implement three methods for gradient boosting:

1. Friedman's original Gradient Boosting Machine (GBM) (Friedman, 2001);
2. Randomized Gradient Boosting Machine (RGBM) (Lu and Mazumder, 2020);
3. Accelerated Gradient Boosting Machine (AGBM) (Lu et al., 2020).

For simplicity, all theory related to the three papers cited above will be analysed using the notation from Lu and Mazumder's 2020 paper (except for section 2.3.4, where we follow the notation of Lu et al., 2020).

2 Gradient Boosting Machine (GBM)

Empirical evidence supports Friedman's 2001 idea, the popularity of which has grown over time: combining an ensemble of *weak* learners leads to better prediction and better generalization results with a testing dataset. In other words, the ensemble of *weak* learners is able to produce a *strong* learner (Friedman, 2001).

Below you can find the pseudocode for Friedman's GBM as formalized in Lu and Mamuzder, 2020. In this pseudocode we can find a slightly different formulation of the problem, which includes K, the set of weak learners. Since Lu and Mazumder find computational guarantees for GBM and RGBM under the same framework, their notation will be used throughout, and convergence of the corresponding algorithm is jointly described in section 2.3.

```
**Algorithm 1**: Gradient Boosting Machine
Initialize with $f_0(x) = 0$
for  m = 0,...,M-1  do:
(1) Compute  pseudoresidual:  $(r^m = -[\frac{\partial l(y_i, f^m(x_i))}{\partial f^m(x_i)}]_{i=1,...,n})$
(2) Find the best weak learner:
$j_m = \arg\min_{j \in [K]} min_\sigma \sum_{i=1}^n (r_i^m - \sigma b(x_i; \tau j))^2$

(3)  Choose the step-size $\rho_m$ by  line  search:
$\rho_m = \arg\min_\rho \sum_{i=1}^n \ l(y_i, f^m(x_i) + \ \rho b(x_i; \tau j_m))^2$

(4)  Update  the  model  $f^{m+1}(x) = f^m(x) + \rho_m b(x_i; \tau j_m))$

**Output**:  $f^M(x)$
```

Friedman's original paper proposes a number of different types of weak learners to be used for the implementation of GBM: least squares regression and regression trees implemented with a LAD (least absolute deviation) loss function, with a Huber loss function, with a

Table 2: Notation

$m = 0, ..., M$	m represents the number of iterations m
(x_i, y_i)	training samples, indexed $i = 1, ..., n$
j_m	best weak learner at iteration m, parameterized by vector τ_j
$[K]$	set $1, ..., K$ of weak learners, of cardinality K

negative binomial log-likelihood loss function and with influence trimming (Friedman, 2001). Regression trees (or stumps) are in fact still today the obvious choice for implementing GBM methods. Refer to Section 3 for more details on our implementation.

The reason GBM and its variants have become such a popular method lies in the fact that, as iterations progress, GBM leads to a sequence of models ${f^m}_{m \in [\mathbf{M}]}$. Each model f^m has its own fit to the data and its own weight, given by the learning rate . Their ensemble, as will be shown in section 2.3, is guaranteed to control the generalization of the model, lowering the variance and guaranteeing that the algorithm will converge "early", i.e. at a set number of iterations or value of the error term (Lu and Mazumder, 2020).

2.1 Randomized Gradient Boosting Machine (RGBM)

From Algorithm 1 it is clear that the most expensive step in GBM is that of finding the best weak learner, since it involves calculating the full gradient at each iteration. However GBM classical implementations often include randomization procedures, for example the column subsampling procedure used in XGBoost (Lu and Mazumder, 2020).

The procedure outlined in Algorithm 2 below is, however, different from the aforementioned randomized procedures. Under a novel theoretical framework, the authors were able to derive the most up-to-date computational guarantees for both GBM and RGBM, which you can find in section 2.3. Moreover, they formalize a new Randomized Gradient Boosting Machine (RGBM) that guarantees significant efficiency improvements over GBM.

```
**Algorithm 2**: Randomized Gradient Boosting Machine
Initialize with $f_0(x) = 0$
for  m = 0,...,M-1  do:
(1) Compute  pseudoresidual: $r^m = -[\frac{\partial l(y_i, f^m(x_i))}{\partial f^m(x_i)}]_{i=1,\ldots,n}$
(2)  Pick a random subset J of weak learners by some rule (Type 0, 1, 2, 3).
(3)  Find the best weak learner: $j_m = \arg\min_{j \in [K]} min_\sigma \ \sum_{i=1}^{n} (r_i^m - \sigma b(x_i; \tau_j))^2$
(4)  Choose the step-size $\rho_m$:
   by line search: $\rho_m = \arg\min_\rho \sum_{i=1}^{n} \ l(y_i, f^m(x_i) + \rho b(x_i; \tau_{jm}))^2$
   constant stepsize:  $\rho_m \ = \rho \ (\sum_{i=1}^{n} \ r_m^i \ b \ (x_i, \tau_{jm}))$
(5)  Update  the  model  $f^{m+1}(x) = f^m(x) + \rho_m b(x_i; \tau j_m))$

**Output**:  $f^M(x)$
```

The basic idea in RGBM is that of lowering the cost of step (3) in Algorithm 2 (corresponding to step (2) in Algorithm 1), by only considering a subset J of the weak learners at each iteration. In the case where the cardinality of the set J is much smaller than the cardinality of the whole set of weak learners, the cost per iteration of RGBM is naturally much lower than that of GBM (Lu and Mazumder, 2020).

Moreover, once a subset of weak leaners has been randomly chosen with one of the selection rules listed below, step (3) of Algorithm 2 finds the coefficients of the best weak learner by applying a greedy coordinate descent rule in the coefficient space. Rewriting the problem in these terms allows the authors to define their procedure as Random-then-Greedy Coordinate Descent (RtGCD), which will be useful for proving convergence in section 2.3 (for proof of equivalence between RGBM and RtGCD in coefficient space see Lu and Mazumder).

- **Type 0 – Full deterministic selection**: Choose J as the whole set of weak learners *

- **Type 1 – Random selection**: Choose uniformly and at random t weak learners from the whole set of weak learners, without replacement. Denote the new subset of weak learners by J.
- **Type 2 – Random single group selection**: Given a nonoverlapping partition of the set of weak learners, pick one group uniformly at random.
- **Type 3 – Random multiple group selection**: Given a nonoverlapping partition of the set of weak learners, pick t groups uniformly at random. Denote the collection of weak learners forming this new subset by J.

* RGBM with Type 0 selection rule is actually classic GBM.

These selection rules share a strong connection with a few types of vector norms, which become important when analyzing the convergence of Algorithms 1 and 2. In particular, details about the relationship between the *infinity norm*, the *ordered l_1 norm* and the *ordered mixed norm*, and Type 0, 1, 2 and 3 selection rules respectively can be found in Lu and Mazumder 2020.

2.2 Accelerated Gradient Boosting Machine (AGBM)

Lu et al.'s 2020 paper incorporates Nesterov's acceleration techniques into Friedman's original GBM. Nesterov showed that the gradient descent (GD) algorithm could be accelerated by adding a *momentum term*: before performing the gradient step, Nesterov's algorithm performs an *extrapolation step* which moves the new iterate along the direction of the difference between the last two iterates. The result of this step is then plugged into the gradient step, eventually bringing GD from a convergence rate of $O(1/\epsilon)$ iterations (proved by Lu and Mazumder, 2020, see section 2.3 for details) to an optimal rate of $O(1/\sqrt{\epsilon})$ iterations for Nesterov's accelerated method. Moreover, Nesterov later demonstrated this rate cannot be improved upon (Nesterov, 2004).

As suggested by the authors, the difficulty in accelerating an algorithm like Friedman's GBM lies in the fact that it uses weak learners: if Nesterov's method was to be applied in a naïve fashion, errors may accumulate in the momentum term (Lu et al., 2020). Therefore, the authors devise a new algorithm for accelerating GBM, the Accelerated Gradient Boosting Machine described by Algorithm 3 below.

AGBM introduces an additional model, h^m, called the *momentum model.* In a real setting where the weak learners are quite simple and it is almost impossible to exactly fit r^m (i.e. the so-called pseudo-residuals), the update of this model is decoupled from the update of the f sequence (see step (4) and (7) Algorithm 3). This prevents the accumulation of error in the *momentum term*, together with the momentum-parameter $\gamma \in (0, 1)$.

Thus, at each iteration we compute two different pseudo-residuals: r^m and c^m (i.e. the so-called corrected residuals). It is useful to note that, when the class of weak learners is *strong* (i.e. exactly fits the residuals), pseudo-residuals and corrected residuals are equal to each other (Lu et al., 2020). The quantity Θ (Minimal Cosine Angle, see section 2.3.2) plays a very significant role in the convergence analysis of the three algorithms presented, and this quantity indeed measures how strong the class of weak learners is by measuring how dense the learners are in the prediction space. Convergence analysis of AGBM can be found in section 2.3.3.

```
Algorithm 3: Accelerated Gradient Boosting Machine
Input: starting function $f^0(x) = 0$, step-size $\eta$,
momentum parameter $\gamma \in (0,1]$, and data $X, y = (x_i, y_i)_{i\in[n]}$

Initialize with $h^0(x) = f^0(x)$ and sequence $\theta_m = \frac{2}{m+2}$
For m = 0,...,M-1 do:
(1) Compute pseudoresidual: $r^m = -[\frac{\partial l(y_i, g^m(x_i))}{\partial g^m(x_i)}]_{i=1,\dots,n}$
(3) Find the best weak learner for pseudoresidual:
$\tau_{m,1} = arg\,min_{\tau\in T} \sum_{i=1}^n (r_i^m - b_\tau(x_i))^2$

(4) Update the model: $f^{m+1}(x) = g^m(x) + \eta\, b_{\tau_{m,1}}(x)$
(5) Update the corrected residual:

$c_i^m = \begin{cases} \mathrm{r}_i^m & \text{if } m = 0 \\ \mathrm{r}_i^m + \frac{m+1}{m+2}\,(c_i^{m-1} - b_{\tau m-1,2}(x_i) & \text{otherwise} \end{cases}$

(6) Find the best weak-learner for the corrected residual:
$\tau_{m,2} = arg\,min_{\tau\in T} \sum_{i=1}^n (c_i^m - b_\tau(x_i))^2$
(7) Update the momentum model: $h^{m+1}(x) = h^m(x) + \frac{\gamma\eta}{\theta_m b_{\tau_{m,2}}(x)}$

Output: $f^M(x)$
```

2.3 Convergence analysis

2.3.1 Loss function: assumptions

Equation (1) below denotes the derivative of the bivariate loss function l with respect to the prediction f. The loss function is, obviously, what we try to minimize in all three algorithms (GBM, RGBM, AGBM) and their convergence is dependent on the loss function respecting the assumptions shown below (assumption (3) is not always necessary, as shown in the analysis).

Loss function: Partial derivative and convexity assumptions
Partial derivative of loss function l wrt prediction f:

$$\frac{\partial l(y, f)}{\partial f} \tag{1}$$

We say that l is σ-smooth if for any y and scalar predictions f_1 and f_2 it holds that:

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

We say that l is μ-strongly convex if for any y and scalar predictions f_1 and f_2 it holds that:

$$l(y, f_1) \leq l(y, f_2) + \frac{\partial l(y, f_2)}{\partial f}\,(f_1 - f_2) + \frac{\mu}{2}\,(f_1 - f_2)^2 \tag{3}$$

The intuition behind these assumptions is that $l(x)$ should not be a function whose gradient changes abruptly (smoothness) and it should also not be completely flat (strong-convexity) (Lu et al., 2020).

Table 3: Convergence rates*

GBM	$O(1/\epsilon)$ *or* $O(1-\frac{\mu}{\sigma}\Theta_{\mathcal{F}}^2)$ *where* $\mathcal{F}$ *is the infintiy vector norm***
RGBM	$(O(1-\frac{\mu}{\sigma}\Theta_{\mathcal{S}}^2)^{\frac{p}{t}}$, *where* $\mathcal{S}$ *is the ordered* l_1 *norm**,* *p is the dimension of the problem and* *t is the number of groups* ($\|K\|$, *see Table 1*)
AGBM	$O(1/\Theta^2\sqrt{\epsilon})$

* as demonstrated in Lu et al., 2020 and Lu and Mazumder, 2020
** for the definition of these norms and their relationship to RGBM group selection rules, see Lu and Mazumder, 2020.

2.3.2 Minimal Cosine Angle

The tool which enables Lu and Mazumder to provide the best available convergence analysis for both GBM and RGBM is a quantity called Minimal Cosine Angle (MCA). This quantity measures the density of the collection of weak learners in the prediction space, thus describing how well the weak learner approximates the desired residual (i.e. the negative gradient of the loss function).

If the learners were strong, the MCA (denoted by Θ) would be equal to 1, indicating that the prediction space is complete; in the case of a space of learners such as tree stumps, we expect this quantity to be much lower, indicating that the prediction space is not very dense (Lu et al., 2020). A definition of MCA can be found below (see Table 2 for notation).

Minimal Cosine Angle: Definition

$$\Theta \; := \; \text{min}_{r\in[R^n]}\text{max}_{\tau\in[T]} \; cos(r, b_\tau(X)) \tag{4}$$

where $b_\tau(X) \in \; R^n$ is a vector of predictions $[b_\tau(x_i)]$

This quantity remains fundamental both for establishing the convergence of GBM and RGBM and for the slightly different approach used for proving convergence of AGBM.

2.3.3 Summary of convergence analysis for GBM and RGBM

In this section we sketch a proof of convergence for GBM and RGBM, both in the case of a strongly convex loss function and of a non-strongly convex one (for the relationship between GBM and RGBM see section 2.1). The convergence analysis of AGBM will be treated separately in section 2.3.4 by virtue of its different structure. It will be useful to compare the classical formulation of the Gradient Boosting Machine minimization problem with the formulation used for deriving the computational guarantees you can find below and in Table 3. Since Lu and Mazumder obtain the best methodological framework for obtaining computational guarantees for both GBM and RGBM, their proof is what has been followed in constructing the following sketch (for the full proof see Lu and Mazumder, 2020).

GBM minimization problem: Let $(x_i, y_i),\ i = 1, ...n$ be training samples s.t. $x_i \in \mathbb{R}^p$ is the feature vector of the i-th sample and y_i is a label (in a classification problem) or a continuous response (in a regression problem).

Let the basis function $b(x;\tau) \in \mathbb{R}$ (=weak learner) be a function of the feature vector indexed by parameter τ, and β_j be the coefficient of the j-th weak learner. Remember K is the cardinality of the finite set of weak learners, by assumption, and M is the total number of iterations of GBM or RGBM.

- **Classical formulation**:
  - Prediction model:
$$f(x) \ := \ \sum_{m=1}^{M} \beta_{jm} b(x; \tau_{jm}) \tag{5}$$
  - Minimization problem:
$$\min_f \ \sum_{i=1}^{n} l(y_i, f(x_i)) \tag{6}$$
- **Formulation in the coefficient space** (i.e. the space of $\beta \in \mathbb{R}^K$):
  - Prediction model:
$$f(x) \ := \ \sum_{j=1}^{K} \beta_j b(x; \tau_j) \tag{7}$$
  - Minimization problem:
$$\min_\beta \ L(\beta), \ where \tag{8}$$
$$L(\beta) \ = \ \sum_{i=1}^{n} l(y_i, \sum_{j=1}^{K} \beta_j b(x_i; \tau_j)) \tag{9}$$

GBM and RGBM computational guarantees

- **Strongly convex loss function**

  Let $\mathbb{E}_m$ denote the expectation over the random selection scheme at iteration m, conditional on the selections up to iteration m-1.

  Let $\mathbb{E}_{\xi_m}$ denote the expectation over the random selection scheme up to and including iteration m.

  THEOREM 1: let l be μ-strongly convex and σ-smooth. For Algorithm 1 and 2 either with line-search or constant step-size rule with $\rho \ = \frac{1}{\sigma}$,
  for all $M \geq 0$ we have:
$$\mathbb{E}_{\xi_M}[L(\beta^M) \ - \ L(\beta^*)] \ \leq \ (1 \ - \ \frac{\mu}{\sigma} \ \Theta_F^2)^M \ (L(\beta^0) \ - \ L(\beta^*)) \tag{10}$$

  Remarks:

- **RGBM**: the multiplicative improvement per epoch (= one epoch is the cost to evaluate learners across all samples) for RGBM is:

$$(1 \ - \ \frac{\mu}{\sigma}\ \Theta_S^2)^{\frac{p}{t}} = \begin{cases} (1 \ - \ \frac{\mu}{p\sigma})^{\frac{p}{t}}) & \text{if } t \geq \sqrt{p} \\ (1 \ - \ \frac{t^2\mu}{p^2\sigma})^{\frac{p}{t}}) & \text{otherwise} \end{cases} \tag{11}$$

- **GBM**: in the special case when J is chosen as the set of all weak learners, Theorem 1 leads to a linear convergence rate for GBM (see Table 3). This is equivalent to saying that, for GBM, $t = p$.

Consider the following propositions, illustrated in Lu and Mazumder, 2020, with:

- matrix B denoting the prediction for all samples over every possible weak learner, namely $B_{i,j} = b(x_i, \tau j),\ for\ i \in [n], j \in [K]$;
- $\mathbb{F}$ denoting the four types of vector norms corresponding to the four selection rules (see section 2.1) and $\mathbb{F}^*$ denoting their dual norms;
- $Dist^B_{F^*}$ is a distance metric in the β space introduced by Lu and Mazumder, 2020. Recall that if $\beta_1, \beta_2 \in Z(\hat{\beta})$, where $Z(\hat{\beta})$ is the invariant subspace of $\hat{\beta}$, then $Dist^B_{\mathbb{F}^*}\ (\beta_1, \beta_2) = 0$

Propositions:
For any iteration index m, we have:

$$\mathbb{E}_m[(\nabla_{jm} L(\beta^m))^2] = ||[\nabla_j L(\beta^m)^2]_j||_F \geq ||\nabla(\beta^m)||^2_F \tag{12}$$

For $a \in \text{Range}(B^T)$ and $t \geq 0$, it holds that:

$$\min_\beta \left\{ \langle\ a, \beta - \beta^* \rangle + \frac{t}{2}\ Dist^B_{F^*}(\beta, \beta^*)^2 \right\} = -\frac{1}{2t}||a||^2_F \tag{13}$$

If l is μ-strongly convex, it holds for any β and $\hat{\beta}$ that:

$$L(\hat{\beta}) \geq L(\beta) + \langle \nabla L(\beta), \hat{\beta} - \beta \rangle + \frac{1}{2}\mu\Theta^2_{F^*} Dist^B_{F*}(\hat{\beta}, \beta) \tag{14}$$

Proof of THEOREM 1: Consider that l is σ-smooth. For either the line-search or the constant step-size rule, it holds that:

$$L(\beta^{m+1}) \leq L(\beta^m) - \frac{1}{\sigma}\nabla_{jm} L(\beta^m) e_{jm}) = L(\beta^m) - \frac{1}{2\sigma}(\nabla_{jm} L(\beta^m))^2 \tag{15}$$

and thus $L(\beta^{m+1}) \leq L(\beta^m$ almost surely. Taking expectations on both sides and using Proposition 12, we get:

$$\mathbb{E}_{m+1}[L(\beta^{m+1})] \leq L(\beta^m) - \frac{1}{2\sigma}||\nabla L \beta^m||^2_F \tag{16}$$

Moreover, it follows from Propositions 13 and 14 that:

$$L(\beta^*) = \min_\beta L(\beta) \geq L(\beta^m) - \frac{1}{2\mu\Theta^2_F}||\nabla L(\beta^m||^2_F \tag{17}$$

Finally, note that (16) together with (17) leads to:

$$\mathbb{E}_{m+1}[L(\beta^{m+1}) - L(\beta^*)] \leq (1 - \frac{\mu}{\sigma}\Theta_F^2)(L(\beta^m) - L(\beta^*)) \tag{18}$$

which concludes the proof.

GBM and RGBM computational guarantees

- **Non-strongly convex loss function**

THEOREM 2: Consider Algorithm 2 with either line-search or constant step-size rule with $\rho = \frac{1}{\sigma}$. If l is a σ-smooth function and has a bounded level set, it holds for all $M \geq 0$ that:

$$\mathbb{E}_{\xi_M}[L(\beta^M) - L(\beta^*)] \leq \frac{1}{\frac{1}{L(\beta^0)-L(\beta^*)} + \frac{M}{2\sigma Dist_0^2}} \leq \frac{2\sigma Dist_0^2}{M} \tag{19}$$

Define the initial level set of the loss function in the β-space as $LS_0 = \left\{\beta \mid L(\beta) \leq L(\beta^0)\right\}$ and its maximal distance to the optimal solution set in $Dist_{F*}^B$ as:

$$Dist_0 = \max_{\beta \in LS_0} Dist_{F*}^B(\beta, \beta^*) \tag{20}$$

Proposition 1: Suppose l has a bounded level set. Then $Dist_0$ is finite.
Remark: LS_0 is bounded in $Dist_{F*}$, i.e. $Dist_0$ is finite when the scalar loss function l has a bounded level set. Recall that by the extreme value theorem, any continuous function on a closed interval is bounded.

Proposition 2: For $a \in \text{Range}(B^T)$, it holds that:

$$||a||_F Dist_{F*}^B(\beta, \hat{\beta}) \geq \langle a, \beta - \hat{\beta}\rangle \tag{21}$$

Proof of THEOREM 2: Recall inequality (16) from Proof of Theorem 1 for a strongly convex loss function, and the fact that $L(\beta^{m+1}) \leq L(\beta^m)$ almost surely. Thus, for any iteration m, with probability one, we have $\beta^m \in LS_0$.

Define $\delta_m := \mathbb{E}_{\xi m}[L(\beta^m) - L(\beta^*)]$.

Using the fact that $\nabla L(\beta^m) \in \text{Range}(B^T)$, convexity of L and Proposition 2 above, we have (see Lu and Mazumder, 2020 for full process):

$$\mathbb{E}_{m+1}[L(\beta^{m+1})] \leq L(\beta^m) - \frac{(L(\beta^m) - L(\beta^*))^2}{2\sigma Dist_0^2} \tag{22}$$

Taking expectation with respect to ξ_m on both sides we arrive at:

$$\mathbb{E}_{\xi m+1}[L(\beta^{m+1})] \leq \mathbb{E}_{\xi m}[L(\beta^m)] - \frac{(\mathbb{E}_{\xi m}[L(\beta^m) - L(\beta^*))])^2}{2\sigma Dist_0^2} \tag{23}$$

By the definition of δ_m we have $\delta_m \geq 0$ and $\delta_{m+1} \leq \delta m - \frac{\delta_m^2}{2\sigma Dist_0^2}$

Noticing that $\delta_{m+1} = \mathbb{E}_{\xi_m}[\mathbb{E}_{m+1}[L(\beta^{m+1}) \mid \xi_m] \leq \mathbb{E}_{\xi_m}[L(\beta^m)] = \delta_m$ and dividing both sides by $\delta_m\delta_{m+1}$, we arrive at:

$$\frac{1}{\delta_{m+1}} \geq \frac{1}{\delta_m} + \frac{1}{2\sigma Dist_0^2} \tag{24}$$

It follows that:

$$\frac{1}{\delta_M} \geq \frac{1}{\delta_0} + \frac{M}{2\sigma Dist_0^2} \tag{25}$$

which completes the proof.

2.3.4 Summary of convergence analysis for AGBM

AGBM computational guarantees

- **Strong convexity parameter μ is unknown**

THEOREM 3: Consider AGBM (Algorithm 3). Suppose l is σ-smooth, the step-size $\eta \leq \frac{1}{\sigma}$ and the momentum parameter $\gamma \leq \frac{\Theta^4}{(4+\Theta^2)}$, where Θ is the MCA introduced in section 2.3.2. Then for all $M \geq 0$, we have:

$$L(f^M) - L(f^*) \ \leq \ \frac{1}{2\eta\gamma(M+1)^2}||f^*(X)||_2^2 \ , \tag{26}$$

where

$$f^* \in \arg\min_{f\in lin(B)} L(f) := \sum_{i=1}^{n} l(y_i, f(x_i)) \tag{27}$$

Proof of THEOREM 3: Recall: $\theta_m = \frac{2}{m+2}$; $r^m = -[\frac{\partial l(y_i, g^m(x_i))}{\partial g^m(x_i)}]_{i=1,\dots,n}$. Introduce α defined as follows:

$$\alpha_m := \frac{\eta\gamma}{\theta_m} = \frac{\eta s\Theta^2}{\theta_m}, \ where \ \ s = \frac{\gamma}{\Theta^2} \tag{28}$$

Define the corrected residuals and the sequence $\hat{h}^{m+1}(X)$ as:

$$c^m = r^m + \frac{\alpha_{m-1}}{\alpha_m}[c^{m-1} - b_{\tau^2_{m-1}}(X)] \tag{29}$$

$$\hat{h}^{m+1}(X) = \hat{h}^m(X) + \alpha_m r^m \tag{30}$$

Where the relationship between $\hat{h}^{m+1}$ and h^{m+1} is given by:

$$\hat{h}^{m+1}(X) = h^{m+1}(X) + \alpha_m(c_m - b_{\tau_{m,2}}(X)) \tag{31}$$

Proof of equation (31) can be found in Lu et al., 2020, Appendix C. The next step is to define the potential function $V(f)$ for any given output function f:

$$V^m(f) = \frac{\alpha_{m-1}}{\theta_{m-1}}(L(f^m) - L(f)) + \frac{1}{2}||f(X) - \hat{h}^m(X)||^2 \tag{32}$$

At every step, the authors show that the potential decreases at least by:
$\delta_m :=\ V^{m+1}(f) \leq V^m(f) + \delta_m$
Where the term δ_m depends on Θ and is defined as:

$$\delta_m := \frac{s\alpha_{m-1}^2}{2t}||c^{m-1} - b_{\tau_{m-1}^2}(X)||^2 - (1-s-t)\frac{\alpha_m^2}{2s}||r^m||^2 \tag{33}$$

By telescope (see Lu et al., 2020, Appendix C, Lemma C.4 for proofs of (33) and (34)) it holds that:

$$\frac{\eta\gamma}{\theta_m^2}(L(f^{m+1}) - L(f^*)) \leq V^{m+1}(f^*) \leq \sum_{j=0}^{m}\delta_j + \frac{1}{2}||f^*(X) - \hat{h}^0(X)||^2 \tag{34}$$

Using tools similar to those used for the convergence analysis of GBM and RGBM (see Lu et al., 2020, Appendix C, Lemma C.5 for details), we are able to prove that, for any given m, it holds that $\sum_{j=0}^{m}\delta_j \leq 0$.
From the facts we just stated, it follows that (see Lu et al., 2020, Appendix C for details) as the term

$$\frac{1}{2}||f^M(X) - f^*(X)||^2 \geq 0, \quad \textit{the term } V^M(f^*) \geq \frac{\alpha_{m-1}}{\theta_{m-1}}(L(f^M) - L(f^*)) \tag{35}$$

After some additional steps, this induces:

$$L(f^M) - L(f^*) \leq \frac{1}{2\gamma\eta}\frac{||f^0(X) - f^*(X)||^2}{M^2} \tag{36}$$

which completes the proof.
Remark: Theorem 3 implies that to get a function f^M such that the error $L(f^M) - L(f^*) \leq \epsilon$, we need a number of iterations $M = O(\frac{1}{\Theta^2\sqrt{\epsilon}})$. This means that for small values of ϵ, AGBM can require far fewer weak learners than GBM.

- **Strong convexity parameter μ is unknown**: this proof is irrelevant to our implementation fo AGBM. For details, see Lu et al., 2020, *Theorem 4.2*.

3 Numerical experiments

This section illustrates the results of the numerical experiments we performed on different datasets in order to quantitatively characterize the behavior of the three boosting machines previously described.

Algorithms 1,2 and 3 were compared following the approach described in Lu et al., 2020 (*GBM vs AGBM*) and Lu, Mazumder, 2020 (*GBM vs RGBM*). Our implementation differs from the authors' original implementation, mostly because of the different programming paradigm used. The implementation details and dissimilarities will be further highlighted throughout the section.

All work presented has been carried out using python programming language.

3.1 GBM vs AGBM

Our implementation of the GBM algorithm follows step by step the pseudocode of *algorithm 1* (see *section 2*), while implementation of the AGBM algorithm follows step by step the pseudocode of *algorithm 3* (see *section 2.2*).

Both algorithms use decision trees (max. depth = 4) as weak learners (*WL*) . Decision trees were implemented with Python package *scikit-learn*. Using this package for the implementation of weak learners prevents the occurrence of any implementation errors in constructing the decision trees. However the use of scikit-learn for WLs constrained us in the manipulation of WL-related hyperparameters, as will be discussed later in this paragraph.

As per the pseudocodes, all ensembles of weak learners were initialized to 0. We adopted a logistic loss function as the loss function for classification problems, and a least-squares loss for regression problems. Differently from Lu et al., 2020 we used all observations to fit the WLs, instead of downsampling the observations to 100 quantiles.

3.1.1 GBM vs AGBM : comparison between different learning rates

The first numerical experiments compare GBM versus AGBM with 3 different levels of the γ hyperparameter: 0.1, 0.3 and 0.5. We conduct this comparison for a classification task (dataset used: *a1a*).

We computed train and test loss in all models with 3 different learning rates ρ: 0.01, 0.1 and 1.

The x-axis values represent the number of WLs included in the ensemble used for prediction(note that this corresponds to the number of iterations as described in the relative pseudocode). Note that, for each iteration of the AGBM algorithm, we fit two WLs: one that will be included in the ensemble used for prediction (i.e. f) , the other that is used as momentum ensemble (i.e. h). This detail is crucial for the interpretation of the plots shown below.

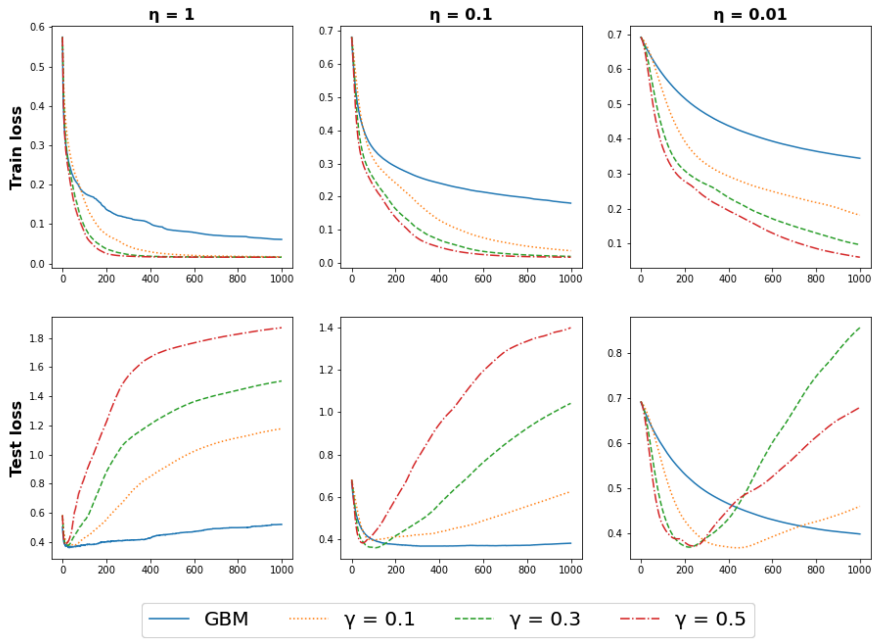
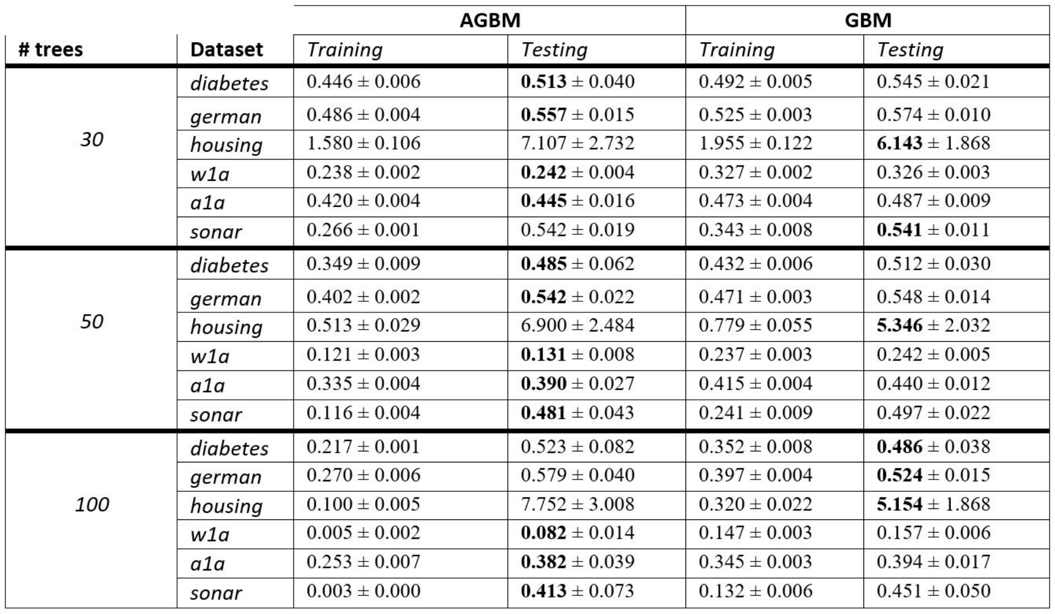


Figure 1: Train (first row) and test (second row) losses for GBM and AGBM (γ: 0.1, 0.3, 0.5) on dataset a1a (classification task). Different learning rates are used ($\rho = 0.01, 0.1, 1$). Losses are plotted as a function of the number of WLs included in the ensemble used for prediction (on the x-axis).

3.1.2 GBM vs AGBM : comparison between different datasets

In the second group of experiments we compare the two methods on multiple datasets, considering 5 classification problems and 1 regression problem. We compare train and test loss between GBM and AGBM on all 6 datasets using different numbers of weak learners: 30, 50 and 100. Due to differences in the implementation of the WLs (i.e. we used scikit-learn, while Lu implements WLs from scratch as python classes) we were not able to use random search with cross validation to find the optimal parameters for the two models (i.e. decision trees specific hyperparameters, plus γ parameter for AGBM). Thus, we fix γ value at 0.3 (i.e. the median value of γ from experiments conducted in sections 3.1.1) and fit the models using the scikit-learn built-in methods for decision trees.

Following the approach used in Lu et al., 2020, in order to obtain better generalization results we trained and tested the two models on 5 different splittings of the datasets. The associated standard error of the mean (SEM) is indicated in Figure 2.



		AGBM		GBM	
# trees	Dataset	*Training*	*Testing*	*Training*	*Testing*
30	*diabetes*	0.446 ± 0.006	**0.513** ± 0.040	0.492 ± 0.005	0.545 ± 0.021
	german	0.486 ± 0.004	**0.557** ± 0.015	0.525 ± 0.003	0.574 ± 0.010
	housing	1.580 ± 0.106	7.107 ± 2.732	1.955 ± 0.122	**6.143** ± 1.868
	w1a	0.238 ± 0.002	**0.242** ± 0.004	0.327 ± 0.002	0.326 ± 0.003
	a1a	0.420 ± 0.004	**0.445** ± 0.016	0.473 ± 0.004	0.487 ± 0.009
	sonar	0.266 ± 0.001	0.542 ± 0.019	0.343 ± 0.008	**0.541** ± 0.011
50	*diabetes*	0.349 ± 0.009	**0.485** ± 0.062	0.432 ± 0.006	0.512 ± 0.030
	german	0.402 ± 0.002	**0.542** ± 0.022	0.471 ± 0.003	0.548 ± 0.014
	housing	0.513 ± 0.029	6.900 ± 2.484	0.779 ± 0.055	**5.346** ± 2.032
	w1a	0.121 ± 0.003	**0.131** ± 0.008	0.237 ± 0.003	0.242 ± 0.005
	a1a	0.335 ± 0.004	**0.390** ± 0.027	0.415 ± 0.004	0.440 ± 0.012
	sonar	0.116 ± 0.004	**0.481** ± 0.043	0.241 ± 0.009	0.497 ± 0.022
100	*diabetes*	0.217 ± 0.001	0.523 ± 0.082	0.352 ± 0.008	**0.486** ± 0.038
	german	0.270 ± 0.006	0.579 ± 0.040	0.397 ± 0.004	**0.524** ± 0.015
	housing	0.100 ± 0.005	7.752 ± 3.008	0.320 ± 0.022	**5.154** ± 1.868
	w1a	0.005 ± 0.002	**0.082** ± 0.014	0.147 ± 0.003	0.157 ± 0.006
	a1a	0.253 ± 0.007	**0.382** ± 0.039	0.345 ± 0.003	0.394 ± 0.017
	sonar	0.003 ± 0.000	**0.413** ± 0.073	0.132 ± 0.006	0.451 ± 0.050

Figure 2: Train-Test losses (last value ± SEM) obtained by GBM and AGBM ($\gamma = 0.3$) on different datasets. In both models a fixed stepsize $\rho = 0.1$ is used. For each model we fit 30, 50 and 100 weak learners.

3.2 GBM vs RGBM

As described in *section 2.1*, the guiding principle behind RGBM is to save computational time by restricting the parameter space when WLs are fitted to pseudo-residuals.

We therefore implemented RGBM following the pseudocode of *algorithm 2*, with type 3 selection rule (for further details see section *2.1*). In order to have comparable results with the experiments previously illustrated in section 3.1, we use a fixed stepsize in our implementation ($\rho = 4$ for classification, ρ=1 for regression).

GBM is just a special case of the aforementioned RGBM. Indeed, when following type 3 selection rule with $t = p$ (recall t = number of features selected for WL fitting, p = total number of features in the dataset) RGBM works as classical GBM.

Thus, in the following comparisons GBM corresponds to the same RGBM implementation described above, with the difference that all features were used to fit the WLs. Moreover, we show that the loss obtained by the implementation of GBM described in the previous section (see *section 3.1*) and by the one described here was the same on the same dataset (*a9a*, data not shown).

As for the previously described models, we use as weak learners decision trees implemented using scikit learn. This time however, following Lu, Mazumder, 2020 , we set tree depth to 1.

As before, the ensembles of weak learners was initialized to 0. Least square loss was adopted for regression problems, while for classification problems we adopted a modified version of the classic logistic loss function as described in Lu and Mazumder, 2020.

We used all observations to fit the WLs instead of downsampling the observations to 100 quantiles, coherently with the previous implementations of GBM and AGBM.

3.2.1 GBM vs RGBM: iterations vs running time

In order to assess the computational advantage we expect RGBM to have with respect to GBM (see *section 2.1* for details), we compare the loss obtained by RGBM with type 3 selection rule using different numbers of features (recall: t=number of features selected; when all features are selected RGBM is equal to GBM). The comparison has been carried out on dataset *a9a* in the context of a classification problem.

In particular we plotted train and test loss against number of weak learners (same as plotting in *section 3.1.1*) and against running time.

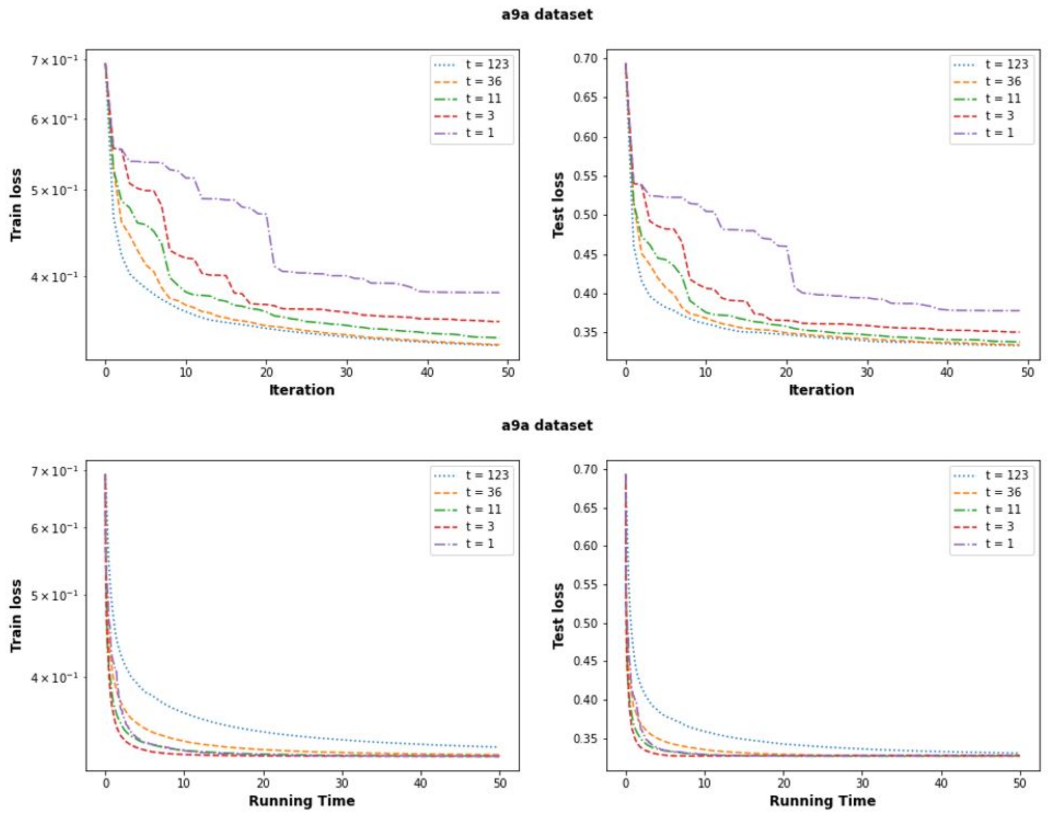


Figure 3: Plots showing loss functions (i.e train and test loss) versus number of RGBM iterations (top panels), and versus running time (in seconds; bottom panels) for RGBM with different t values. Different values of t corresponds to different cardinalities of the feature set used for model fitting (note: highest t corresponds always to total feature set (i.e. GBM)).

3.2.2 GBM vs RGBM: running time with different datasets

Finally, we analyze the convergence results of GBM and RGBM using four different datasets (3 were used for classification tasks and 1 for a regression task). We used Type 3 selection rule in the implementation of RGBM also in this case, obtaining different random subsets of features. Results were reported by plotting train and test loss against running time.

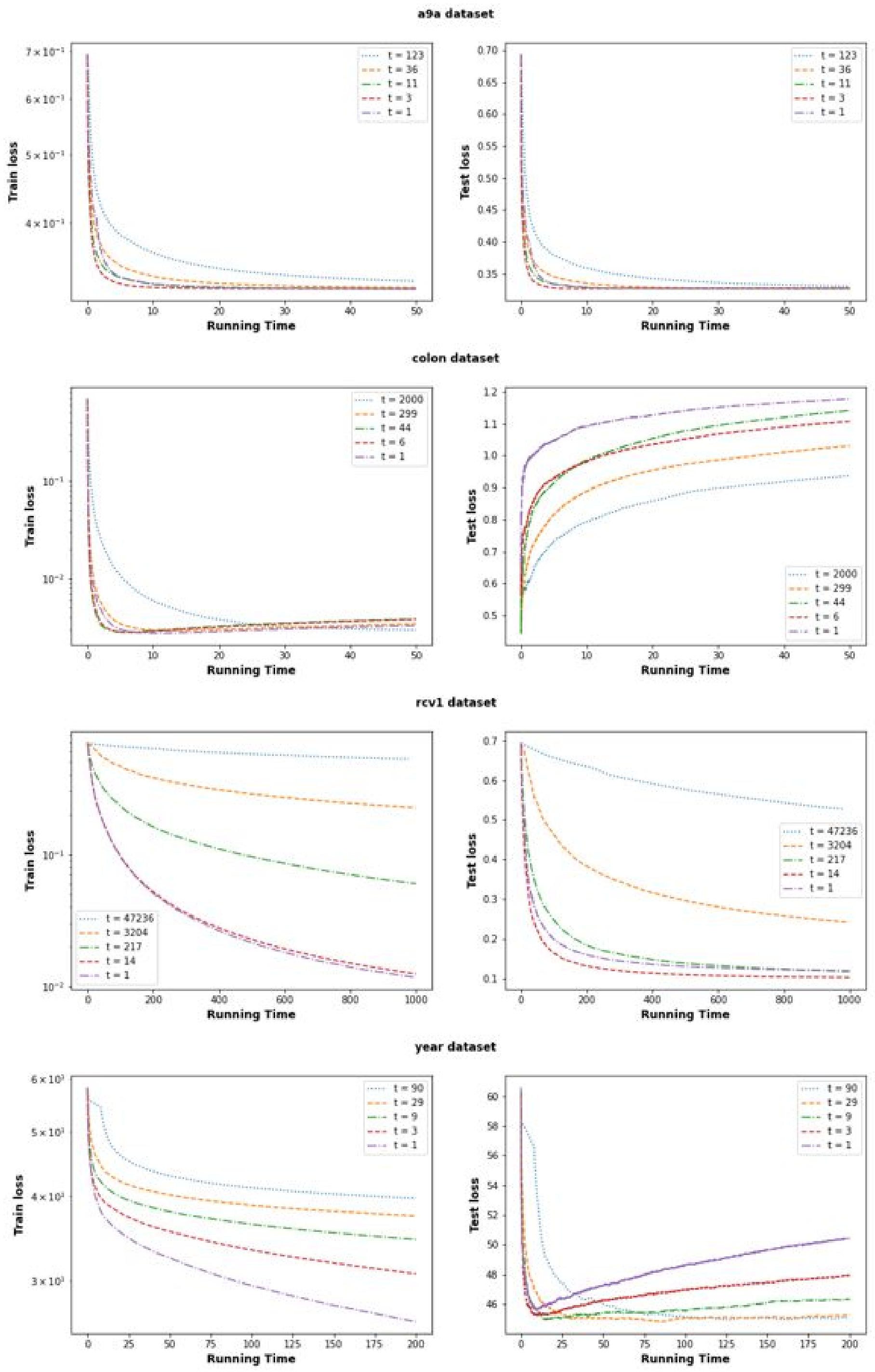


Figure 4: Plots showing RGBM train and test loss values plotted versus running time (in seconds) for different datasets. Different t values are used (note: highest t corresponds always to total feature set (i.e. GBM)).

4 Discussion

The main goal of our project was to compare the three methods presented and implemented so far. We conducted several experiments on different datasets, both for classification and regression problems.

The comparisons of GBM against AGBM and GBM against RGBM were treated separately, since we wanted to underline different results: AGBM has a significantly better convergence rate but similar computational cost compared to GBM, while the strength of RGBM lies in its much cheaper cost per iteration.

4.1 Comparison: GBM vs AGBM

We can observe from Figure 1 that AGBM method implemented with different values of γ makes the train loss converge faster than classical GBM. Thus, AGBM method reaches minimum test loss with less iterations compared to GBM. However, AGBM tends to overfit with a higher number of iterations. This issue can be easily overcome by the implementation of early stopping procedures, as suggested in Lu et al., 2020.

Comparing our implementation with the original Lu et al. 2020 implementation we observe slight differences in the test losses. In particular our AGBM models tend to reach the minimum test loss value in less iterations and clearly show a stronger overfitting behavior. This might be motivated by the fact that we use all the observations instead of extracting their quantiles as Lu et al. choose to do for their implementation.

From Figure 2 we can observe that, for a small number of weak learners (iterations), AGBM clearly outperforms GBM on different datasets. The trend observed in our losses clearly follows the one reported in Lu et al. 2020. Numerical differences could be imputed to previously described differences from the original implementation. In Lu et al. 2020 in particular, the authors optimize several hyperparameters of the decision tree models (including γ). Because of our choice of using built-in models from the scikit-learn package, we were not able to follow their implementation.

4.2 Comparison: GBM vs RGBM

Coherently with the theory, we can observe from Figure 3 that RGBM where weak learners are fitted using less features (Type 3 feature selection rule) converges faster than GBM in terms of running time, although GBM converges faster in both training and testing loss in terms of iterations. From this we can derive that, by optimizing the t hyperparameter (i.e. the number of features used to fit the weak learner), we can reach a sufficiently satisfying solution using less computational resources compared to classical GBM. The faster convergence of RGBM is reinforced by Figure 4 where similar results are obtained on multiple datasets.

Our results do however show major differences with the results on specific datasets analyzed in Lu et Mazumder 2020. We do not have a coherent interpretation for the reasons behind these differences.

5 Bibliography

Friedman, J.H., 2001. Greedy function approximation: a gradient boosting machine. Annals of statistics, pp.1189-1232.

Lu, H., Karimireddy, S.P., Ponomareva, N. and Mirrokni, V., 2020, June. Accelerating Gradient Boosting Machines. In International Conference on Artificial Intelligence and Statistics (pp. 516-526). PMLR.

Lu, H. and Mazumder, R., 2020. Randomized gradient boosting machine. SIAM Journal on Optimization, 30(4), pp.2780-2808.

Nesterov, Y., 2004. Introductory lectures on convex optimization: A basic course. Springer Science Business Media, volume 87.