Proximal boosting and its acceleration

Erwan Fouillen, Claire Boyer, Maxime Sangnier

Proximal boosting and its acceleration

Erwan Fouillen[1], Claire Boyer[1,2], and Maxime Sangnier[1,3]

[1]Sorbonne Université, CNRS, LPSM, Paris, France
[2]Ecole Normale Supérieure, CNRS, DMA, Paris, France
[3]Sorbonne Université, CNRS, LIP6, Paris, France

January 22, 2020

Abstract

Gradient boosting is a prediction method that iteratively combines weak learners to produce a complex and accurate model. From an optimization point of view, the learning procedure of gradient boosting mimics a gradient descent on a functional variable. This paper proposes to build upon the proximal point algorithm when the empirical risk to minimize is not differentiable to introduce a novel boosting approach, called proximal boosting. Besides being motivated by non-differentiable optimization, the proposed algorithm benefits from Nesterov's acceleration in the same way as gradient boosting [Biau et al., 2018]. This leads to a variant, called accelerated proximal boosting. Advantages of leveraging proximal methods for boosting are illustrated by numerical experiments on simulated and real-world data. In particular, we exhibit a favorable comparison over gradient boosting regarding convergence rate and prediction accuracy.

1 Introduction

Boosting is a celebrated machine learning technique, both in statistics and data science. In broad outline, boosting combines simple models (called weak learners) to build a more complex and accurate model. This assembly is performed iteratively, taking into account the performance of the model built at the previous iteration. The way this information is considered leads to several variants of boosting, the most famous of them being Adaboost [Freund and Schapire, 1997] and gradient boosting [Friedman, 2001].

The reason of the success of boosting is twofold: i) from the statistical point of view, boosting is an additive model with an iteratively growing complexity. In this sense, boosting lies between ensemble methods (which aggregate weak learners) and strong models (such as nonparametric ones). In practice, it combines the best of both worlds by reducing the variance and the bias of the risk; ii) from the data science perspective, boosting is a model, the fitting of which is computationally cheap. In contrast, it can quickly achieve highly complex models, thus it is able to perform accurately on difficult learning task. As an ultimate feature, the iterative process makes finding the frontier between under and overfitting quite easy. In particular, gradient boosting combined with decision trees (often referred to as gradient tree boosting) is currently regarded as one of the best off-the-shelf learning techniques in data challenges.

As explained by Biau et al. [2018], gradient boosting has its roots in Freund and Schapire's work on combining classifiers, which resulted in the Adaboost algorithm [Schapire, 1990, Freund, 1995, Freund and Schapire, 1996, 1997]. Later, Friedman and colleagues developed a novel boosting procedure inspired by the numerical optimization literature, and nicknamed gradient boosting [Friedman et al., 2000, Friedman, 2001, 2002]. Such a connection of boosting between statistics and optimization was already stated in several previous analyses by Breiman [Breiman, 1997, 1998, 1999, 2000, 2004] and

reviewed as functional optimization [Mason et al., 2000b,a, Meir and Rätsch, 2003, Bühlmann and Hothorn, 2007]: boosting can be seen as an optimization procedure (similar to gradient descent), aimed at minimizing an empirical risk over the set of linear combinations of weak learners. In this respect, a few theoretical studies prove the convergence, from an optimization point of view, of boosting procedures [Zhang, 2002, 2003, Wang et al., 2015] and particularly of gradient boosting [Temlyakov, 2012, Biau and Cadre, 2017]. Let us remark that rates of convergence of gradient boosting are known for strongly convex and strongly smooth risks [Rätsch et al., 2002, Grubb and Bagnell, 2011].

It is quite surprising that in gradient boosting (and variants), the number of weak learners controls both the number of optimization steps performed in order to minimize the empirical risk and the statistical complexity of the final predictor. The former feature, consisting in stopping the optimization algorithm before convergence by choosing adequately the number of iterations, is known, in many areas, as early stopping. It can be seen as an iterative regularization mechanism used to prevent overfitting [Lin et al., 2016]. As a consequence, besides the numerical learning procedure of gradient boosting, its statistical performance deeply relies on the algorithm employed. Especially as early stopping operates jointly with another regularization mechanism: the control of the model complexity.

That being said, one may wonder if gradient descent is really a good option. Following this direction, several alternatives have been proposed, such as replacing gradient descent by the Frank-Wolfe algorithm [Wang et al., 2015], incorporating second order information [Chen and Guestrin, 2016], and applying Nesterov's acceleration [Biau et al., 2018]. While all these variants rely on differentiable loss functions, Grubb and Bagnell [2011] discuss the limitations of boosting with gradient descent in the non-differentiable setting, and tackle these issues by proposing two modified versions of (sub)gradient boosting based on projections of accumulated (sub)gradients on the set of weak learners. The contribution of the work described here is to go a step forward by proposing a procedure to efficiently learn boosted models with non-differentiable loss functions, and with a potential acceleration feature (such as accelerated gradient boosting [Biau et al., 2018]).

To go into details, Section 2 reviews boosting with respect to the empirical risk minimization principle and illustrates the flaw of the current learning procedure in a simple non-differentiable case: least absolute deviations. Then, some backgrounds on non-smooth optimization are stated in Section 3 and we explain the main contribution of this paper: adapting the proximal point algorithm [Nesterov, 2004] (and its acceleration) to boosting. The proposed method is nicknamed proximal boosting (respectively accelerated proximal boosting). A second contribution is the derivation of the weights of weak learners for accelerated descents (including accelerated gradient boosting [Biau et al., 2018]). While we empirically observe that accelerated boosting may diverge, Section 4 addresses convergence of non-accelerated proximal boosting: theoretical convergence guarantees are stated (based on the work by Rockafellar [1976]) and a convergence rate is established in a restrictive but representative setting. Finally, the numerical study described in Section 5 shines a light on advantages and limitations of the proposed boosting procedures.

2 Problem and notation

Let $\mathcal{X}$ be an arbitrary input space and $\mathcal{Y} \subseteq \mathbb{R}$ an output space. Given a pair of random variables $(X, Y) \in \mathcal{X} \times \mathcal{Y}$, supervised learning aims at explaining Y given X, thanks to a function $f_0 \colon \mathcal{X} \to \mathbb{R}$. In this context, $f_0(X)$ may represent several quantities, depending on the task at hand, for which the most notable examples are the conditional expectation $x \in \mathcal{X} \mapsto \mathbb{E}(Y|X = x)$ and the conditional quantiles of Y given X for regression, as well as the regression function $x \in \mathcal{X} \mapsto \mathbb{P}(Y = 1|X = x)$ for ± 1-classification. Often, this target function f_0 is minimizer of the risk $\mathbb{E}(\ell(Y, f(X)))$ over all integrable functions f, where $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a suitable convex loss function (respectively the square function and the pinball loss in the regression examples previously mentioned).

Since the distribution of (X, Y) is generally unknown, the minimization of the risk is out of reach. One would rather deal with its empirical version instead. Let $\{(X_i, Y_i)\}_{1 \leq i \leq n} \subseteq \mathcal{X} \times \mathcal{Y}$ be a training sample of pairs (X_i, Y_i) independent and identically distributed according to the distribution of (X, Y)

and $\mathcal{F} \subseteq L^2(\mu_X)$ a class of functions integrable for the distribution μ_X of X. In this work, we consider estimating f_0 by means of an additive model $f^\star$ (that is $f^\star = \sum_{t=0}^{T} w_t g_t$, where T is an unknown integer and $(w_t, g_t)_t \subseteq \mathbb{R} \times \mathcal{F}$ is an unknown sequence of weights and weak learners) by solving the following optimization problem:

$$\underset{f \in \operatorname{span} \mathcal{F}}{\text{minimize}} \; C(f), \tag{P1}$$

where

$$C(f) = \mathbb{E}_n(\ell(Y, f(X))) = \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, f(X_i))$$

is the empirical risk and $\operatorname{span} \mathcal{F} = \{\sum_{t=1}^{m} w_t g_t : w \in \mathbb{R}^m, (f_1, \ldots, f_m) \in \mathcal{F}^m, m \in \mathbb{N}\}$ is the set of all linear combinations of functions in $\mathcal{F}$ ($\mathbb{N}$ being the set of non-negative integers).

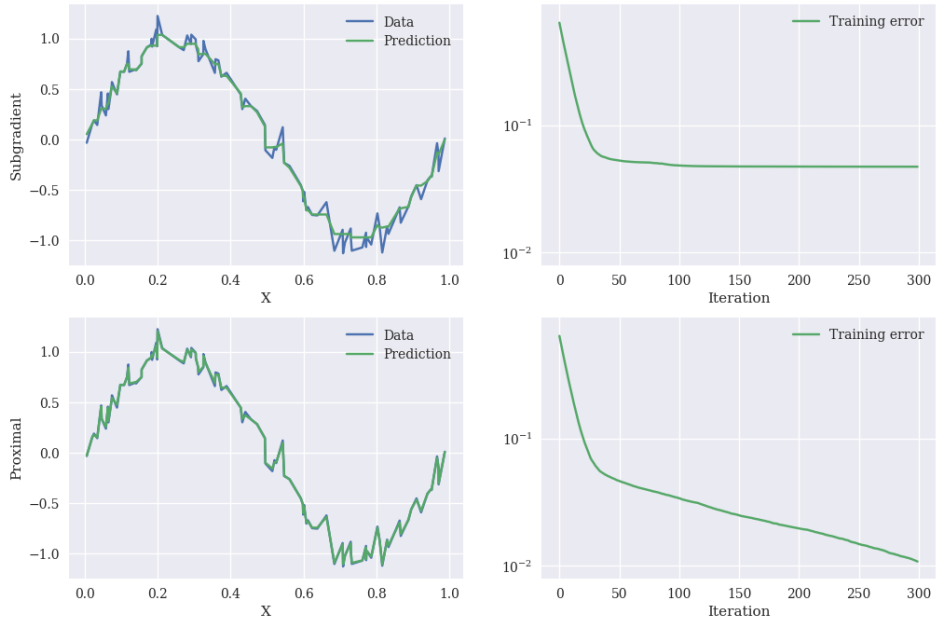

Figure 1: Predicted values and training error of a boosting machine trained with a subgradient (top) and a proximal-based method (bottom).

As a simple example, let us consider the regression model $Y = \sin(2\pi X) + \epsilon$, where X is uniformly distributed on $[0, 1]$ and ϵ is normally distributed and independent of X. We aim at solving:

$$\underset{f \in \operatorname{span} \mathcal{F}}{\text{minimize}} \; \mathbb{E}_n(|Y - f(X)|),$$

with $\mathcal{F}$ being the set of regression trees of depth less than 3.

Two boosting machines $f_T = \sum_{t=0}^{T} w_t g_t$ are learned (with T fixed to 300): a traditional one with a subgradient-type method, and another with the proposed proximal-based procedure. Figure 1 depicts the prediction of f_T (left) and the training error $C(f_t) = \mathbb{E}_n(|Y - f_t(X)|)$ along the iterations t (right).

In an optimization perspective, it appears clearly that the subgradient method fails to minimize the empirical risk (prediction is far from the data and the training error is stuck fare above 10^{-2}) while the proximal-based procedure constantly improves the objective. The subgradient method faces a flaw in convergence, in all likelihood due to non-differentiability of the absolute function $|\cdot|$. This simple example illustrates, inside the boosting paradigm, a well-known fact in numerical optimization: proximal-based algorithms prevails over subgradient techniques.

3 Algorithm

There is an ambiguity in (P1), since it is a functional optimization problem but, in practice, we do not necessarily have the mathematical tools to apply standard optimization procedures (in particular concerning differentiation of C). For this reason, C is often regarded as a function from $\mathbb{R}^n$ to $\mathbb{R}$, considering that it depends on f only through $(f(X_i))_{1\leq i\leq n}$. To make this remark more precise, let, for all $z \in \mathbb{R}^n$, $D(z) = \frac{1}{n}\sum_{i=1}^n \ell(Y_i, z_i)$. Then, it is enough to remark that for any $f \in L^2(\mu_X)$, $C(f) = D(f(X_1^n))$, where $f(X_1^n) = (f(X_1), \dots, f(X_n)) \in \mathbb{R}^n$ is the vector of f computed on the training sample.

Having this remark in mind helps solving (P1), for instance considering that taking the gradient of C with respect to f is roughly equivalent to differentiating C with respect to $f(x)$ (for all observed $x \in \{X_1, \dots, X_n\}$), thus taking in fact the usual gradient of D. Doing so, the only requirement is to juggle functions appearing in Problem (P1) and their vectorial twins coming from the optimization procedure. In particular, given a vectorial gradient $\nabla D(f(X_1^n))$ ($f \in \mathrm{L}^2(\mu_X)$), one has to find a function $g \in \mathrm{L}^2(\mu_X)$ that correctly represents it, i.e. such that $g(X_1^n) \approx \nabla D(f(X_1^n))$. This principle is at the heart of functional optimization methods such that the ones used in boosting [Mason et al., 2000b].

From now on, all necessary computations of C with respect to f, can be forwarded to D. For instance, if ℓ is differentiable with respect to its second argument, we can define, for all $f \in L^2(\mu_X)$, the functional gradient of C as $\nabla_{\mathrm{n}} C(f) = \nabla D(f(X_1^n))$. On the contrary, if ℓ is not differentiable, we may consider a subgradient of C at f, denoted $\widetilde{\nabla}_{\mathrm{n}} C(f)$ and defined as any subgradient of D at $f(X_1^n)$.

In the forthcoming sections, a common first order optimization algorithm is reviewed. Then, it is explained how to build different procedures for solving (P1), according to the properties of the loss function ℓ.

3.1 Accelerated proximal gradient method

Let us assume for a while that we want to minimize the function $g + h$, where $g\colon \mathbb{R}^d \to \mathbb{R}$ is convex and differentiable (with L-Lipschitz continuous gradient, $L > 0$), and $h\colon \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ is convex and lower semi-continuous. Besides, let us define the proximal operator of h by:

$$\mathrm{prox}_h(x) = \arg\min_{u\in\mathbb{R}^d} h(u) + \frac{1}{2}\|u - x\|_{\ell_2}^2, \qquad \forall x \in \mathbb{R}^d.$$

For any point $x \in \mathbb{R}^d$, $\mathrm{prox}_h(x)$ is well defined by convexity and low semi-continuity of h [Combettes and Wajs, 2005]. Then, the iterative procedure defined by choosing any $x_0 = v_0 \in \mathbb{R}^d$ and by setting for all $t \in \mathbb{N}$:

$$\begin{cases} x_{t+1} = \mathrm{prox}_{\gamma h}(v_t - \gamma \nabla g(v_t)) \\ v_{t+1} = x_{t+1} + \alpha_{t+1}(x_{t+1} - x_t) \end{cases}$$

where $\gamma \in (0, 1/L]$ and $(\alpha_t)_t$ will be made precise thereafter, is known to converge to a minimizer of $g+h$ [Nesterov, 2004]. The rate of convergence depends on the choice of α_t: if $\alpha_t = 0$ for all $t \in \mathbb{N}$, then the previous procedure leads to the well known proximal gradient method, which converges in $O(1/t)$. More formally, assuming that $g + h$ has a minimizer $x^\star$, then $(g + h)(x_t) - (g + h)(x^\star) = O(1/t)$. On the other hand, if one chooses the sequence $(\alpha_t)_t$ defined recursively by:

$$\begin{cases} \beta_0 = 0 \\ \beta_{t+1} = \frac{1+\sqrt{1+4\beta_t^2}}{2}, t \in \mathbb{N} \\ \alpha_{t+1} = \frac{\beta_t - 1}{\beta_{t+1}}, t \in \mathbb{N}, \end{cases} \tag{1}$$

then the convergence becomes $O(1/t^2)$. This is in the spirit of the acknowledged acceleration proposed by Nesterov [1983], and generalized to the composite setting by Beck and Teboulle [2009].

Depending on the properties of the objective function to minimize, the procedure described before leads to two simple algorithms (coming with their acceleration):

- the gradient method ($h = 0$):
$$x_{t+1} = v_t - \gamma \nabla g(v_t),$$
minimizes a single function g as soon as it is convex and differentiable with Lipschitz-continuous gradient;

- the proximal point algorithm ($g = 0$):
$$x_{t+1} = \text{prox}_{\gamma h}(v_t) = v_t - \gamma \left[\frac{1}{\gamma} \left(v_t - \text{prox}_{\gamma h}(v_t) \right) \right], \tag{2}$$
minimizes a single function h, which is only required to be convex and lower semi-continuous (in that case, there is no restriction on the step size γ, except being positive). Let us remark that, using the recursive update of v_t, x_{t+1} can also be expressed by:
$$x_{t+1} = x_t + \alpha_t (x_t - x_{t-1}) + \gamma \left[\frac{1}{\gamma} \left(\text{prox}_{\gamma h}(v_t) - v_t \right) \right], \tag{3}$$
which will be useful for designing the proposed algorithm.

Without acceleration (i.e. with $\alpha_t = 0$, for all $t \in \mathbb{N}$), the proximal gradient method (as well as its two special cases) has the asset to be a descent method: at each iteration, the objective function monotonically decreases, meaning that $(g + h)(x_{t+1}) \leq (g + h)(x_t)$, with convergence rate at least $O(1/t)$ ($O(1/t^2)$ with Nesterov's acceleration). In particular, this is true when minimizing a single convex and lower semi-continuous function $h \colon \mathbb{R}^d \to \mathbb{R}$, even if it is not differentiable.

This has to be put in contrast with the subgradient method:
$$x_{t+1} = x_t - \gamma_t \widetilde{\nabla} h(x_t), \tag{4}$$
where $\gamma_t > 0$ and $\widetilde{\nabla} h(x_t)$ is any subgradient of h at x_t. This procedure, which is very similar to the gradient descent but replacing the gradient by any subgradient, has a convergence rate $O(1/\sqrt{t})$ in the best case (that is, when γ_t is well chosen) [Nesterov, 2004]. In addition, this rate is optimal for this optimization procedure: it cannot be improved without extra assumptions on h [Nesterov, 2004, Theorem 3.2.1]. This means that there does not exist an acceleration scheme for this approach.

This remark motivates the use of procedures different from the subgradient method when minimizing a non-differentiable function h, such as the proximal point algorithm described in Equation 2. This motivation is emphasized by the fact that moving from subgradient to proximal point method only requires to replace the update direction $\widetilde{\nabla} h(x_t)$ by $\frac{1}{\gamma}(x_t - \text{prox}_{\gamma h}(x_t))$ (see Equations (4) and (2), keeping in mind that $v_t = x_t$ in the non-accelerated setting). This observation is the cornerstone of the algorithm proposed in Section 3.3.

3.2 Gradient boosting

Let $\mathcal{F}_0$ be the set of constant functions on $\mathcal{X}$ and assume that $\mathcal{F}_0 \subseteq \mathcal{F}$. Also, for any $f \in L^2(\mu_X)$, let us denote $\|f\|_{\mu_n} = \sqrt{\mathbb{E}_n(f(X)^2)} = \sqrt{\frac{1}{n}\sum_{i=1}^n f(X_i)^2}$. Then, a simple procedure to approximately solve (P1) is gradient boosting, described in Algorithm 1 [Mason et al., 2000a, Friedman, 2001]. It builds the requested additive model in an iterative fashion, by imitating a gradient (or subgradient if ℓ is not differentiable with repect to its second argument) method. At each iteration t, Algorithm 1 finds a function g_{t+1} that approximates the opposite of a subgradient of C (also called pseudo-residuals) and adds it to the model f_t with a weight $w_{t+1} = \nu \gamma_{t+1}$, where $\nu \in (0, 1]$ is a shrinkage coefficient (also called learning rate) and $\gamma_{t+1} \in \mathbb{R}$ is the optimal step given the direction g_{t+1}. At the end of the procedure, the proposed estimator of f_0 is $f_T = \sum_{t=0}^T w_t g_t$, with $w_0 = 1$.

Algorithm 1 requires a number of iterations T, which acts on two regularization mechanisms. The first one is statistical (T controls the complexity of the subspace in which f_T lies) and the second

Algorithm 1 Gradient boosting.

Input: $\nu \in (0, 1]$ (shrinkage coefficient).

1: Set $f_0 = g_0 \in \arg\min_{g \in \mathcal{F}_0} C(g)$ (initialization).

2: **for** $t = 0$ **to** $T - 1$ **do**

3: Compute $r \leftarrow -\widetilde{\nabla}_{\mathrm{n}} C(f_t)$ (pseudo-residuals).

4: Compute

$$g_{t+1} \in \arg\max_{g \in \mathcal{F}, \|g\|_{\mu_n} \leq 1} \langle g(X_1^n), r \rangle_{\ell_2} \qquad \text{(correlation)}$$

or

$$g_{t+1} \in \arg\min_{g \in \mathcal{F}} \|g(X_1^n) - r\|_{\ell_2}. \qquad \text{(least squares)}$$

5: Compute

$$\gamma_{t+1} \in \arg\min_{\gamma \in \mathbb{R}} C(f_t + \gamma g_{t+1})$$

6: Set $f_{t+1} = f_t + \nu \gamma_{t+1} g_{t+1}$. (update).

7: **end for**

Output: f_T.

one is numerical (T controls the precision to which the empirical risk C is minimized). The shrinkage coefficient ν tunes the balance between these two regularization mechanisms.

Algorithm 1 has two variants according to the way the subgradient of C is approximated (respectively by correlation or by least squares). The first one closely relates to AdaBoost [Mason et al., 2000a] while the second one is officially known as gradient boosting [Friedman, 2001].

Let us remark that the line search (Line 5 of Algorithm 1) simply scales the weak learner g_t by a constant factor. However, when the class $\mathcal{F}$ is a set of regression trees, g_t is a piecewise constant function. In this case, it is common to perform a line search sequentially for each leaf of the decision tree [Friedman, 2001] (called a multiple line search). As a consequence, each level of the piecewise constant function g_t is scaled with its own factor.

3.3 Boosting with non-differentiable loss functions

When the function ℓ is not differentiable with respect to its second argument, gradient boosting just uses a subgradient $\widetilde{\nabla}_{\mathrm{n}} C(f_t)$ instead of the gradient $\nabla_{\mathrm{n}} C(f_t)$. This is, of course, convenient but as explained previously, far from leading to interesting convergence properties. For this reason, we propose a new procedure for non-differentiable loss functions ℓ, which consists in adapting the proximal point algorithm [Nesterov, 2004] to functional optimization.

For any $f \in L^2(\mu_X)$, let $\psi_\lambda C(f) = \frac{1}{\lambda}\left(f(X_1^n) - \mathrm{prox}_{\lambda D}(f(X_1^n))\right)$, where $\lambda > 0$ is a parameter. The simple idea underlying the proposed algorithm, nicknamed (accelerated) proximal boosting, is that the only difference between subgradient and proximal point methods are the update directions of the optimization variable, which are respectively $\widetilde{\nabla}_{\mathrm{n}} C(f_t)$ and $\psi_\lambda C(f_t)$. Thus, proximal boosting computes the pseudo-residuals based on $\psi_\lambda C(f_t)$ instead of $\widetilde{\nabla}_{\mathrm{n}} C(f_t)$. In addition, this iterative procedure can be sped up by applying Nesterov's acceleration, reviewed in Section 3.1.

The accelerated proximal boosting procedure is described in Algorithm 2. It is very similar to Algorithm 1, except that the pseudo-residuals are now given by a proximal operator instead of a subgradient, and that Nesterov's acceleration is available.

Following the acceleration scheme, after approximating the direction of optimization $r \in \mathbb{R}^n$ by g_{t+1}, the iterate update (Line 11 in Algorithm 2) becomes (see Equation (3)):

$$f_{t+1} = f_t + \alpha_t(f_t - f_{t-1}) + \nu \gamma_{t+1} g_{t+1}.$$

Moreover, we have to maintain the vectorial twin $x_{t+1} = f_{t+1}(X_1^n)$ of f_{t+1} as well as an auxiliary

variable $v_t \in \mathbb{R}^n$.

Similarly to regular gradient boosting (Algorithm 1), the estimator returned at the end of Algorithm 2 can be written $f_T = \sum_{t=0}^{T} w_t g_t$, where the weights $(w_0, \ldots, w_T)$ are now slightly more complicated (this is explained in the next section).

Algorithm 2 Accelerated proximal boosting.

Input: $\nu \in (0, 1]$ (shrinkage coefficient), $\lambda > 0$ (proximal step).

1: Set $g_0 \in \arg\min_{g \in \mathcal{F}_0} C(g)$ (initialization).
2: $x_0 \leftarrow g_0(X_1^n) \in \mathbb{R}^n$ (predictions).
3: $v_0 = x_0$ (interpolated point).
4: $(w_0^{(0)}, \ldots, w_T^{(0)}) \leftarrow (1, 0, \ldots, 0)$ (weights of weak learners).
5: **for** $t = 0$ **to** $T - 1$ **do**
6: Compute $r \leftarrow \frac{1}{\lambda} \left(\mathrm{prox}_{\lambda D}(v_t) - v_t \right)$ (pseudo-residuals).
7: Compute

$$g_{t+1} \in \arg\max_{g \in \mathcal{F}, \|g\|_{\mu_n} \leq 1} \langle g(X_1^n), r \rangle_{\ell_2} \qquad \text{(correlation)}$$

or

$$g_{t+1} \in \arg\min_{g \in \mathcal{F}} \|g(X_1^n) - r\|_{\ell_2} . \qquad \text{(least squares)}$$

8: Compute

$$\gamma_{t+1} \in \arg\min_{\gamma \in \mathbb{R}} C(f_t + \gamma g_{t+1}).$$

9: Set $x_{t+1} \leftarrow v_t + \nu \gamma_{t+1} g_{t+1}(X_1^n)$ (which corresponds to $x_{t+1} = f_{t+1}(X_1^n)$).
10: Set $v_{t+1} \leftarrow x_{t+1} + \alpha_{t+1}(x_{t+1} - x_t)$.
11: Set $f_{t+1} \leftarrow f_t + \alpha_t(f_t - f_{t-1}) + \nu \gamma_{t+1} g_{t+1}$ (which corresponds to updating weights $(w_0^{(t+1)}, \ldots, w_{t+1}^{(t+1)})$ according to Equation (5)).
12: **end for**

Output: $f_T = \sum_{t=0}^{T} w_t^{(T)} g_t$.

Let us remark that the idea of applying Nesterov's acceleration to boosting originally appeared in [Biau et al., 2018] for a gradient-type procedure. Even though Biau et al. [2018] did not suggest to apply such an acceleration scheme when ℓ is not differentiable, this idea appears natural. However, this is not entirely relevant since it contradicts the optimization theory (see Section 3.1): subgradient method cannot be accelerated. This flaw clearly motivates using proximal-based methods for non-differentiable boosting, as proposed in Algorithm 2.

3.4 Weights with Nesterov's acceleration

As an additive model, it is of interest to express f_T with respect to the base learners $(g_0, \ldots, g_T)$ and their weights: $f_T = \sum_{t=0}^{T} w_t g_t$. On the first hand, in the non-accelerated case ($\alpha_t = 0$ for all $t \in \mathbb{N}$), the update rule of Algorithm 1 is simply $f_{t+1} = f_t + \nu \gamma_{t+1} g_{t+1}$. Therefore the weights are defined by $w_0 = 1$ and $w_t = \nu \gamma_t$ for all positive integers t.

On the other hand, when Nesterov's acceleration is on ($(\alpha_t)_{t \in \mathbb{N}}$ is defined by Equation (1)), the update rule becomes (see Line 11 in Algorithm 2):

$$f_{t+1} = (1 + \alpha_t) f_t - \alpha_t f_{t-1} + \nu \gamma_{t+1} g_{t+1},$$

which makes the final weights a bit more complicated to obtain. Property 1 (proved in Appendix A) gives the closed-form expression of the weights of f_T in this case.

Property 1. *The weights of f_T are:*

$$\begin{cases} w_0 = 1 \\ w_1 = \nu\gamma_1 \\ w_t = \left(1 + \sum_{j=t}^{T-1} \prod_{k=t}^{j} \alpha_k\right) \nu\gamma_t, \forall t \in \{2, \dots, T-1\} \\ w_T = \nu\gamma_T. \end{cases}$$

In addition, Property 2 (proved in Appendix A) provides a recursive update for implementing Algorithm 2. Let us remark that, Property 2 is also valid for accelerated gradient boosting as proposed by Biau et al. [2018]. This paves the way of efficient implementations of both accelerated gradient and proximal boosting.

Property 2. *Let us denote, for each iteration $t \in \{1, \dots, T-1\}$, $f_t = \sum_{j=0}^{t} w_j^{(t)} g_j$ the expansion of f_t. Then, the weights can be updated according to the following recursion:*

$$\begin{cases} w_0^{(0)} = 1 \\ w_1^{(0)} = \nu\gamma_1 \\ w_1^{(1)} = \nu\gamma_1 \\ w_j^{(t+1)} = (w_j^{(t)} - w_j^{(t-1)})(1 + \alpha_t) + w_j^{(t-1)}, \forall j \in \{1, \dots, t\} \\ w_{t+1}^{(t+1)} = \nu\gamma_{t+1}. \end{cases} \tag{5}$$

4 Convergence results

This section is dedicated to the theoretical convergence of the proximal boosting algorithm, presented in Algorithm 2, in its non-accelerated version. Thus, we assume, in the whole section, that the $(\alpha_t)_t$'s are set to zero, which implies in particular that $v_t = x_t$ at each iteration. The study of the accelerated version will not be covered in this paper and is left as future work.

A preliminary result on the convergence of the proximal boosting technique can be easily derived upon previous work by Rockafellar: it requires the control of the error introduced by considering an approximated direction of optimization instead of the true proximal step, and could be stated as follows.

Theorem 3 ([Rockafellar, 1976, Theorem 1])**.** *Let $(f_t)_t$ be any sequence generated by Algorithm 2 and define for any iteration t:*

$$\varepsilon_{t+1} = \left\| g_{t+1}(X_1^n) - (-\psi_\lambda C(f_t)) \right\|_{\ell_2}.$$

Suppose that $\{x_t\}_t$ is bounded and that

$$\sum_{t=0}^{+\infty} \varepsilon_t < +\infty. \tag{6}$$

Then,

$$\lim_{t \to \infty} C(f_t) = \inf_{f \in \operatorname{span} \mathcal{F}} C(f).$$

Theorem 3 states that as soon as the approximation errors $(\varepsilon_t)_t$ converge to 0 quicker than $1/t$, then the sequence $(C(f_t))_t$ converges to a minimum of C. However, with a better control of the approximation errors $(\varepsilon_t)_t$, a rate of convergence can be derived. This is the role of the result we provide now. It is based on assumptions similar to the theoretical analysis of gradient boosting

available in Grubb and Bagnell [2011]. In particular, the analysis relies on two critical properties of the objective functional C.

A functional C, which admits gradients $\nabla_{\mathrm{n}} C$, is said L-strongly smooth (for some $L > 0$) if for all $f, f' \in L^2(\mu_X)$:

$$C(f') \leq C(f) + \langle \nabla_{\mathrm{n}} C(f), f' - f \rangle_{\mu_n} + \frac{L}{2} \left\| f' - f \right\|_{\mu_n}^2 ,$$

and κ-strongly convex (for some $\kappa > 0$) if

$$C(f') \geq C(f) + \langle \nabla_{\mathrm{n}} C(f), f' - f \rangle_{\mu_n} + \frac{\kappa}{2} \left\| f' - f \right\|_{\mu_n}^2 ,$$

where $\langle f, f' \rangle_{\mu_n} = \mathbb{E}_{\mu_n}(f(X)f'(X)) = \frac{1}{n}\sum_{i=1}^n f(X_i)f'(X_i)$. Let us remark that such properties are directly inherited from the loss function ℓ.

A convergence rate for proximal boosting can be derived from these two properties. It is stated hereafter and proved based on a result presented in Appendix B.

Theorem 4. *Assume that C is L-smooth and κ-strongly convex for some $L > 0$ and $\kappa > 0$. Let $(f_t)_t$ be any sequence generated by Algorithm 2 and assume that there exists $\zeta \in (0, 1]$ such that for any iteration t:*

$$\left\| g_{t+1}(X_1^n) - (-\psi_\lambda C(f_t)) \right\|_{\ell_2}^2 \leq (1 - \zeta^2) \left\| \psi_\lambda C(f_t) \right\|_{\ell_2}^2 . \tag{7}$$

Let $f^\star \in \arg\min_{f \in \operatorname{span} \mathcal{F}} C(f)$ (well defined by strong convexity and linearity of $\operatorname{span} \mathcal{F}$), and choose $\lambda = \frac{\zeta^2}{8L}$. Then,

$$C(f_T) - C(f^\star) \leq \left(1 - \frac{\zeta^2 \kappa}{9L}\right)^T \left(C(f_0) - C(f^\star)\right) .$$

Proof. Given that $\forall f \in L^2(\mu_X) : C(f) = D(f(X_1^n))$, strongly smoothness and convexity assumptions directly translate to the function $x \mapsto nD(x)$ with the same parameters. Thus, the previous result is a straightforward application of Theorem 5 applied to the function $x \mapsto nD(x)$. □

Theorem 4 states that proximal boosting has a linear convergence rate under strongly smoothness and convexity assumptions. This result is similar to the one obtained for gradient boosting in Grubb and Bagnell [2011] and is indeed based on the same assumptions. In particular, the way of controlling the approximation error is common in the boosting literature: when Equation (7) is verified, the set of weak learners $\mathcal{F}$ is said to have edge ζ.

We admit that strongly smoothness and convexity are restrictive assumptions for an algorithm designed for non-differentiable loss functions. However, they seem necessary to control the impact of the approximation error on the convergence. In addition, Section 5 will show that linear convergence (as stated by Theorem 4) is always observed in practical cases (even though the loss is not differentiable).

5 Numerical analysis

In Section 3, proximal boosting (Algorithm 2) has been introduced in a fairly general way. However, following the success of gradient boosting, the empirical results presented in this section only relate to a least squares approximation of the pseudo-residuals, with decision trees (implemented in Scikit-learn [Pedregosa et al., 2011]) of depth at most 3 as base learners (class $\mathcal{F}$) and a multiple line search.

In the whole section, the four methods involved in the numerical comparison are nicknamed:

Gradient (slow) : gradient boosting [Friedman, 2001];

Gradient (fast) : accelerated gradient boosting [Biau et al., 2018];

Proximal (slow) : Algorithm 2 without acceleration ($\alpha_t = 0$, for all $t \in \mathbb{N}$);

Proximal (fast) : Algorithm 2 with Nesterov's acceleration (α_t defined by Equation (1)).

5.1 Impact of parameters on algorithm behaviors

This section aims at numerically illustrating, based on synthetic data, the performance of our proximal boosting algorithm and at highlighting the benefits of coupling Nesterov's acceleration scheme with proximal boosting. For this purpose, two synthetic models are studied (see description below), both coming from [Biau et al., 2016, 2018]. The other models considered in [Biau et al., 2018] have also been studied but results are not reported because they are very similar to the two models we focus on.

Regression : $n = 800, d = 100, Y = -\sin(2X_1) + X_2^2 + X_3 - \exp(-X_4) + Z_{0,0.5}$.

Classification : $n = 1500, d = 50, Y = 2\mathbf{1}_{X_1+X_4^3+X_9+\sin(X_{12}X_{18})+Z_{0,0.1}>0.38} - 1$, where $\mathbf{1}$ is the indicator function.

The first model covers an additive regression problem, while the second covers a binary classification task with covariate interactions. In both cases, we consider an input random variable $X \in \mathbb{R}^d$, the covariate of which, denoted $(x_j)_{1\leq j\leq d}$, are normally distributed with zero mean and covariance matrix $\Sigma = \left(2^{-|i-j|}\right)_{1\leq i,j\leq d}$ (this is the correlated design). We present, in Appendix C, a variant of this framework based on covariates uniformly distributed over $(-1,1)^d$ (the uncorrelated design) and for which the numerical results are identical. Moreover, in these synthetic models of regression and classification, an additive and independent noise (normally distributed with mean $\mu \in \mathbb{R}$ and variance σ^2) is embodied by the random variable Z_{μ,σ^2}.

Four different losses are considered (see Table 1 for a brief description): least squares and least absolute deviations for regression; exponential (with $\beta = 1$) and hinge for classification. Computations for the corresponding (sub)gradients and proximal operators are detailed in Appendix D. On that occasion, it can be remarked that the direction of descent $\psi_\lambda C(f_t)$ of proximal boosting applied with the least squares loss is the same as that of gradient boosting, $\widetilde{\nabla}_{\mathrm{n}} C(f_t)$, up to a constant factor (see Appendix D). In other words, proximal and gradient boosting are exactly equivalent.

In addition, note that we also considered other kind of losses such as the pinball loss for regression and the logistic loss for classification (see Table 1). Nevertheless, since the numerical behaviors are respectively very close to the least absolute deviations and exponential cases, the results are not reported.

Loss	Parameter	$\ell(y, y')$	Type
least squares	-	$(y-y')^2/2$	regression
least absolute deviations	-	$\lvert y-y'\rvert$	regression
pinball	$\tau \in (0,1)$	$\max(\tau(y-y'), (\tau-1)(y-y'))$	regression
exponential	$\beta > 0$	$\exp(-\beta yy')$	classification
logistic	-	$\log_2(1+\exp(-yy'))$	classification
hinge	-	$\max(0, 1-yy')$	classification

Table 1: Loss functions.

In the following numerical experiments, the random sample generated based on each model is divided into a training set (50%) to fit the method and a validation set (50%). The performance of the methods are appraised through several curves representing the training and validation losses along the $T = 5000$ iterations with which the algorithms are run.

5.1.1 Maximal tree depth

As a first numerical experiment, we aim at illustrating Theorem 4 for two classes $\mathcal{F}$ of weak learners: regression trees with maximal depth 3 (in blue in Figure 2) and 15 (in red in Figure 2). This last class of weak learners is supposed to make almost no error in approximating the directions of descent, thus leading to quasi-standard optimization algorithms.

To complete this analysis, we also study the behavior of an algorithm described in Grubb and Bagnell [2011] (referred to as GB's residual). The variant of gradient boosting proposed by Grubb and Bagnell has been designed to handle non-differentiable losses with a convergence in $O(\sqrt{t})$. At each iteration, it considers a negative subgradient completed by the error produced at the previous iteration by the tree approximating the direction of descent.

Since Theorem 4 does not address Nesterov's acceleration (and Grubb and Bagnell's method may not necessarily be accelerated), it is the same for the numerical experiment of this section. In addition, parameters λ and ν are set to standard values: $\lambda = 1$, $\nu = 10^{-2}$, which does not hurt the generality of the forthcoming interpretations.

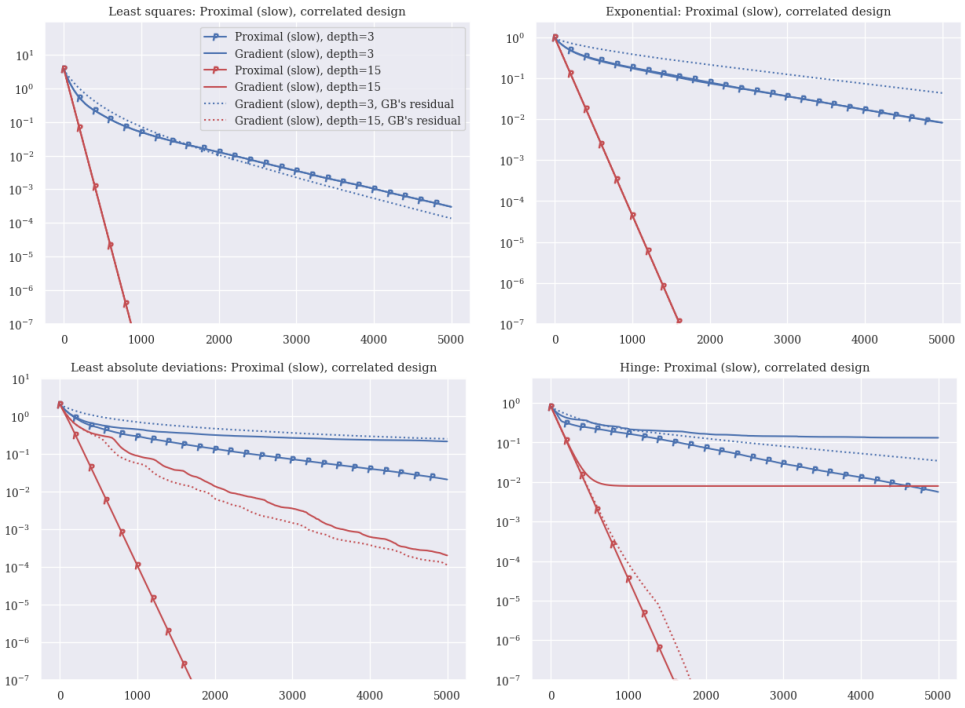

Figure 2: Training losses for two values of maximal depth (3 in blue, 15 in red) and Grubb and Bagnell's residual (dashed lines) – number of iterations on the horizontal axis.

Let us analyze the top panels of Figure 2: for differentiable losses (least squares and exponential), gradient and proximal descents behave exactly the same (curves with and without the symbol **P** are mixed up). Moreover, as theoretically analyzed in Theorem 4, the rate of convergence of proximal boosting is linear with a slope that increases with the capacity of the class of weak learners. As we can see, this result is similar to what is known for gradient boosting [Grubb and Bagnell, 2011].

Let us note that Grubb and Bagnell's residual (the dotted lines in Figure 2) does not seem to help convergence neither with a large class of weak learners (in red, the residual is in fact always almost null), nor with a restricted class (in blue).

It is interesting to observe that even though the last two losses (least absolute deviations and hinge) do not satisfy the hypotheses of Theorem 4, the convergence of proximal boosting is still almost linear with a slope increasing with the capacity of the class of weak learners. On the contrary, gradient boosting behaves very badly compared to our proposed algorithm, particularly for the hinge loss. In this case, Grubb and Bagnell's residual improves the convergence, which comes close to proximal boosting for sufficiently powerful decision trees.

Keeping in mind the behavior of Grubb and Bagnell's residual for gradient boosting (and that it

may not necessarily be accelerated), we leave it behind us for the rest of the numerical convergence analysis and we will get it back when studying the generalization ability.

5.1.2 Learning rate

This subsection tackles the impact of the learning rate parameter $\nu \in (0, 1]$ on the relative performances of both proximal and accelerated proximal methods. Throughout, the proximal step λ is fixed to 1. The convergence rates for $\nu \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ are illustrated in the case of the regression model (least squares and least absolute deviations losses) and in the case of the classification model (exponential and hinge losses) in Figure 3. The numerical results depicted on these two figures are analyzed in the next paragraphs.

Both without and with Nesterov's acceleration, the convergence rate highly depends on the learning rate ν. As one might expect, a higher learning rate leads to a faster convergence.

Besides, the convergence rate is deeply improved by Nesterov's acceleration, even though, accelerated boosting may suffer from divergence, particularly for high values of ν. This is not a real defect since, in practice, divergence occurs in the overfitting regime (after that the minimum validation loss is achieved, as represented by crosses in Figure 3).

Let us remark that oscillations, appearing for accelerated boosting in Figure 3, are not surprising because (contrarily to what its name seems to indicate) Nesterov's accelerated gradient descent is not a descent method, even though it converges faster than standard gradient descent.

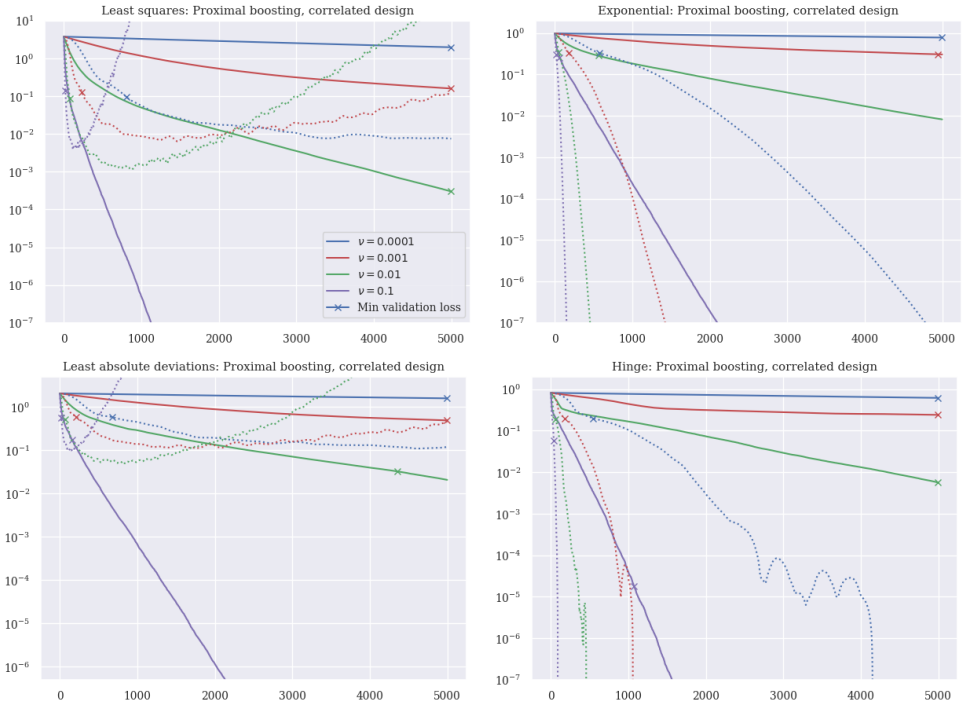

Figure 3: Training losses of proximal boosting (solid line) and its accelerated version (dashed line) for several values of ν – number of iterations on the horizontal axis. For each curve, the iteration where the minimum validation loss is achieved is represented by a cross.

5.1.3 Proximal step

After studying the behavior of (accelerated) proximal boosting with respect to the learning rate ν, we now fix $\nu = 10^{-2}$ (which is a representative value) and study the effect of the proximal step $\lambda > 0$.

In addition (and without loss of generality with respect to the value of the learning rate), standard gradient and accelerated gradient boosting are added to the comparison (still with $\nu = 10^{-2}$).

The convergence rates for $\lambda \in \{10^{-2}, 10^{-1}, \ldots, 10^{2}\}$ are illustrated in Figure 4 both for the regression and the classification models. The numerical results are analyzed in the forthcoming paragraphs.

Differentiability The effect of the proximal step λ is very different according to the nature of the loss. On the first hand, for differentiable losses, λ has almost no effect on the convergence trend, as well exemplified by the exponential loss (remember that for the least squares losses, proximal boosting is equivalent to gradient boosting, so λ has exactly no effect all the curves are mixed up). This holds true both for regular and accelerated proximal boosting and, in this case, numerical results are similar to that of (accelerated) gradient boosting. A direct consequence is that there seems to have no particular advantage in favor of proximal boosting, compared to gradient boosting.

On the other hand, for non-differentiable losses (least absolute deviations and hinge), first, the proximal step has a clear impact on the convergence rate: the bigger λ, the faster the convergence of the training loss. Second, for an adequate value of λ, (accelerated) proximal boosting clearly outperforms (accelerated) gradient boosting. In this situation (when using non-differentiable losses), proximal boosting is really advantageous compared to gradient boosting.

Convergence Similarly to what has been observed previously, Nesterov's acceleration always improves the convergence rate of the training and the validation errors. However, it is important to remark that divergence of accelerated proximal boosting occurs independently of the value of λ. This behavior seems to be only related to the learning rate ν. Again, divergence of accelerated proximal and gradient boosting is not a drawback since it always occurs in the overfitting regime.

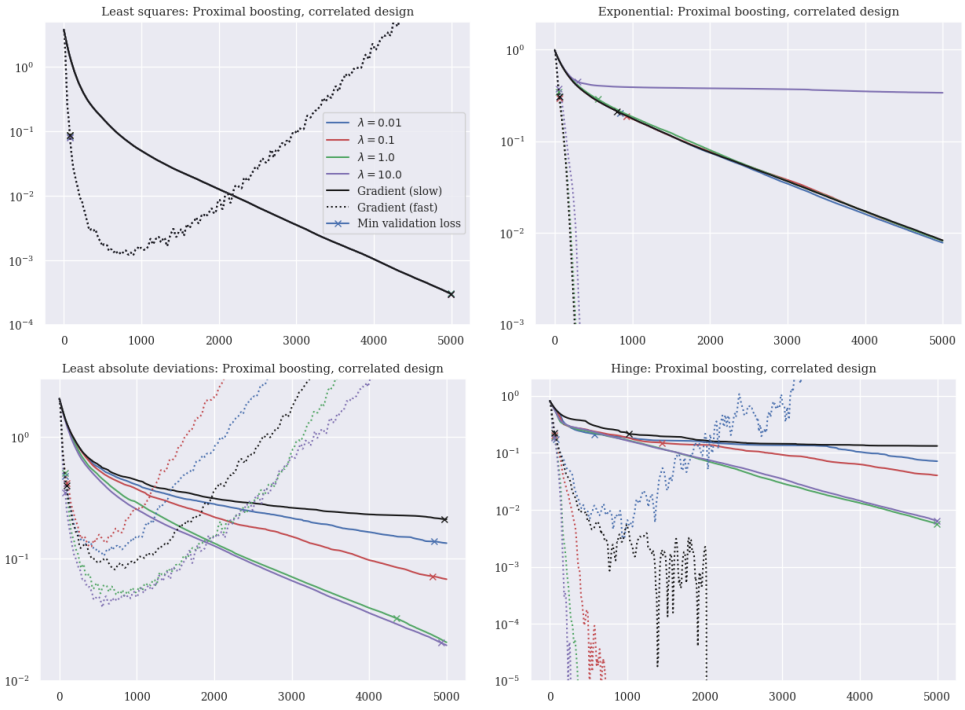

Figure 4: Training losses of proximal boosting (solid line) and its accelerated version (dashed line) for several values of λ – number of iterations on the horizontal axis. For each curve, the iteration where the minimum validation loss is achieved is represented by a cross.

5.1.4 Generalization ability

The goal of this section is to illustrate the generalization ability of the various estimators. For this purpose, we present an optimistic estimator of the generalization ability, computed on the synthetic datasets as the minimal validation error with respect to parameters ν, λ and the maximal depth of decision trees varying in $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$, $\{10^{-2}, 10^{-1}, \ldots, 10^{2}\}$ and $\{1, 3, 5\}$ respectively. Table 2 reports the average (and standard deviations) of this minimal validation error over 20 replications of the synthetic datasets. The method nicknamed *Gradient (GB)* refers to Grubb and Bagnell's residual.

		Slow					Fast				
Loss	Method	Depth	T	ν	λ	Error	Depth	T	ν	λ	Error
Least squares	Gradient	1	5290	3e-02	-	**0.64** (0.24)	1	384	1e-02	-	0.72 (0.26)
	Gradient (GB)	1	4794	6e-02	-	**0.64** (0.23)					
	Proximal	1	4496	5e-02	3	**0.64** (0.24)	1	370	1e-02	2	0.71 (0.26)
Least absolute deviations	Gradient	1	7926	1e-01	-	0.84 (0.05)	2	390	3e-02	-	0.93 (0.06)
	Gradient (GB)	1	8936	1e-01	-	0.85 (0.07)					
	Proximal	1	3147	6e-02	4	**0.77** (0.05)	1	240	3e-02	6	0.84 (0.05)
Exponential	Gradient	1	636	9e-02	-	0.45 (0.03)	1	450	2e-02	-	0.45 (0.03)
	Gradient (GB)	3	2020	2e-02	-	0.46 (0.03)					
	Proximal	1	325	1e-01	6e-01	**0.44** (0.03)	1	396	3e-02	4e-01	0.45 (0.03)
Hinge	Gradient	3	589	9e-02	-	0.32 (0.03)	2	231	4e-02	-	0.30 (0.03)
	Gradient (GB)	3	137	1e-01	-	0.36 (0.02)					
	Proximal	2	2342	7e-02	1	0.32 (0.02)	2	453	1e-02	4	**0.29** (0.03)

Table 2: Generalization ability: average minimal validation errors (with standard deviations) and associated parameters. For each block, boldface values are smallest errors.

Globally, we observe that generalization is enhanced by tree stumps rather than by deeper decisions trees. Moreover, proximal boosting and its accelerated version do not help for the least squares loss, as their average validation errors are similar or greater than vanilla gradient boosting (Table 2, see also Table 4 for the uncorrelated design). However, proximal boosting definitely improves generalization for the three other losses (particularly for least absolute deviations), compared to gradient boosting and Grubb and Bagnell's modification.

Last but not least, accelerated proximal boosting generally provides competitive models but with very few weak learners, as exemplified by the least absolute deviations and hinge losses in Table 2. In addition, let us remark that accelerated proximal boosting always outperforms accelerated gradient boosting [Biau et al., 2018].

5.1.5 Intermediate conclusion

To conclude, the convergence rate of proximal boosting, as well as its generalization ability, depend both on the shrinkage coefficient ν and the proximal step λ (the latter is important mainly for non-differentiable losses).

Moreover, with accelerated boosting, the size of the best boosting model is impacted by ν but not clearly by λ. This is also true for the potential divergence of the training loss, which, again, is not a defect since it always occurs in the overfitting regime.

All in all, the learning rate ν and the proximal step λ have a noticeable impact on the behavior of (accelerated) proximal boosting. Consequently, these parameters have to be carefully tuned.

As an additional piece of conclusion, Nesterov's acceleration always provides far smaller boosting models (compared to without acceleration) with a comparable generalization ability and even a clear improvement when using the hinge loss.

Eventually, compared to (accelerated) gradient boosting, (accelerated) proximal boosting always prevails (regarding convergence and generalization) for non-differentiable losses and offers similar performances for differentiable losses.

5.2 Comparison in real-world cases

This section aims at assessing, on real-world datasets, the generalization ability and the size of the final model for the proposed approaches (see Section 3.3) in comparison to known variants. Intuitively, Algorithm 2 is supposed to behave better than gradient-type boosting when the loss function ℓ is not differentiable. Benefits are expected on the generalization ability (proximal methods are able to minimize $C(f)$, with $f \in \operatorname{span} \mathcal{F}$, as much as one wants) and on the number of iterations (or weak learners) necessary for producing accurate predictions (weak learners $(g_t)_t$ are more likely to minimize C if they are based on proximal methods).

Comparison is based on four datasets (available on the UCI Machine Learning repository), for which the characteristics are described in Table 3. The first two are univariate regression datasets, while the three others relate to binary classification problems. In both situations, the sample is split into a training set (50%), a validation set (25%) and a test set (25%). The parameters of the methods (number of weak classifiers $T \in [1, 2000]$, maximal depth of decision trees varying in $[1, 3, 5]$, learning rate $\nu \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and proximal step $\lambda \in \{10^{-3}, 10^{-2}, \ldots, 10^{2}\}$) are selected as minimizers of the loss computed on the validation set for models fitted on the training set. Then, models are refitted on the training and the validation sets with selected parameters. Finally, the generalization ability of the methods are estimated by computing the loss (and the misclassification rate for classification models) on the test set. These quantities are reported through statistics computed on 20 random splits of the datasets.

Dataset	n	d	Type
crime	1994	101	regression
wine	1599	11	regression
adult	30162	13	classification
advertisements	2359	1558	classification
spam	4601	57	classification

Table 3: Real-world datasets.

The losses considered in these experiments are least squares, least absolute deviations and pinball (with $\tau = 0.9$) for the regression problems, as well as exponential (with $\beta = 1$) and hinge for the classification tasks (see Table 1 for a quick definition and Appendix D for the details). Since random forests [Breiman, 2001] and extreme gradient boosting [Chen and Guestrin, 2016] are competitive aggregating methods designed for least squares regression and classification, they have also been included in the numerical analysis as benchmarks. The parameters of random forests (except the number of trees fixed to 2000 since the more, the better) and extreme gradient boosting (nicknamed *XGBoost* and parameterized with the exponential loss for classification) are selected in accordance with the evaluation procedure described above.

5.2.1 Regression problems

Test losses for the least squares (left), least absolute deviations (middle) and pinball (right) losses are described in Figure 5, along with the number of weak learners selected in Figure 6. First of all, let us remark that the numbers of weak learners (alternatively referred to as model sizes) can be compared across methods since for a given pair task/dataset, the maximal depth tree selected is roughly the same for all methods (as exemplified in Figure 12).

Regarding the least squares setting, let us remind that gradient and proximal boosting boil down to be the same method. We observe that they achieve a precision comparable to extreme gradient

boosting and better than that of random forests. Moreover, accelerated versions of boosting methods do not produce more accurate models than vanilla boosting but with much less weak learners (which is a great advantage, already observed in Biau et al. [2018]). A possible explanation concerning the lower precision of accelerated boosting is that convergence is so fast, that tuning parameters becomes very tricky.

Looking now at least absolute deviations and pinball losses, we observe that proximal boosting achieves a better precision than gradient boosting (particularly for the pinball loss) with equally sized or smaller final models.

In comparison to Grubb and Bagnell's method, proximal boosting achieves comparable precisions on the three tasks but with final models that tend to be smaller.

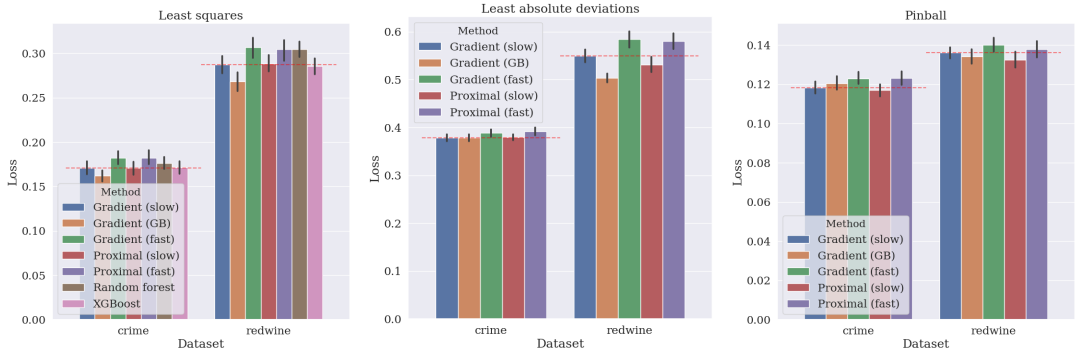

Figure 5: Losses on the test datasets for the least squares (left), least absolute deviations (middle) and pinball (right) losses.

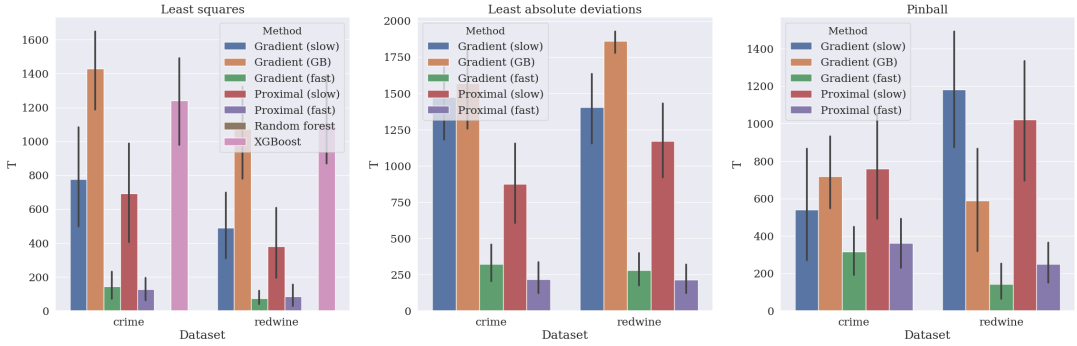

Figure 6: Selected number of weak learners for the least squares (left), least absolute deviations (middle) and pinball (right) losses.

For the two non-differentiable loss functions considered here (least absolute deviations and pinball losses), the good performances of subgradient-based boosting could be explained by the following reason: in this numerical experiment, we are interested in the test loss, which does not require minimizing the training loss entirely. Thus, even though subgradient techniques may not converge, the decrease of the empirical loss may be sufficient for achieving good generalization performances.

5.2.2 Classification problems

Losses (left) and misclassification rates (right) computed on the test dataset are depicted in Figure 7 for exponential (top) and hinge (bottom) losses. Figure 8 shows that the selected number of weak learners for both losses.

One can observe that for the exponential loss, gradient and proximal boosting achieve comparable errors and misclassification rates but outperform random forests and extreme gradient boosting. In comparison, Grubb and Bagnell's method performs sometimes slightly better than proximal boosting, but the size of its final model (as well as that of gradient boosting, extreme gradient boosting and random forests) is always bigger than than of proximal boosting.

For the non-differentiable hinge loss, proximal boosting, accompanied by Grubb and Bagnell's method, clearly outperform gradient-based techniques, concerning both the loss value and the misclassification rate. This is in agreement with what was expected. In addition, as already observed in the synthetic numerical experiment, this is a situation where Nesterov's acceleration improves the performance of gradient and proximal boosting models, accelerated proximal boosting being the more accurate.

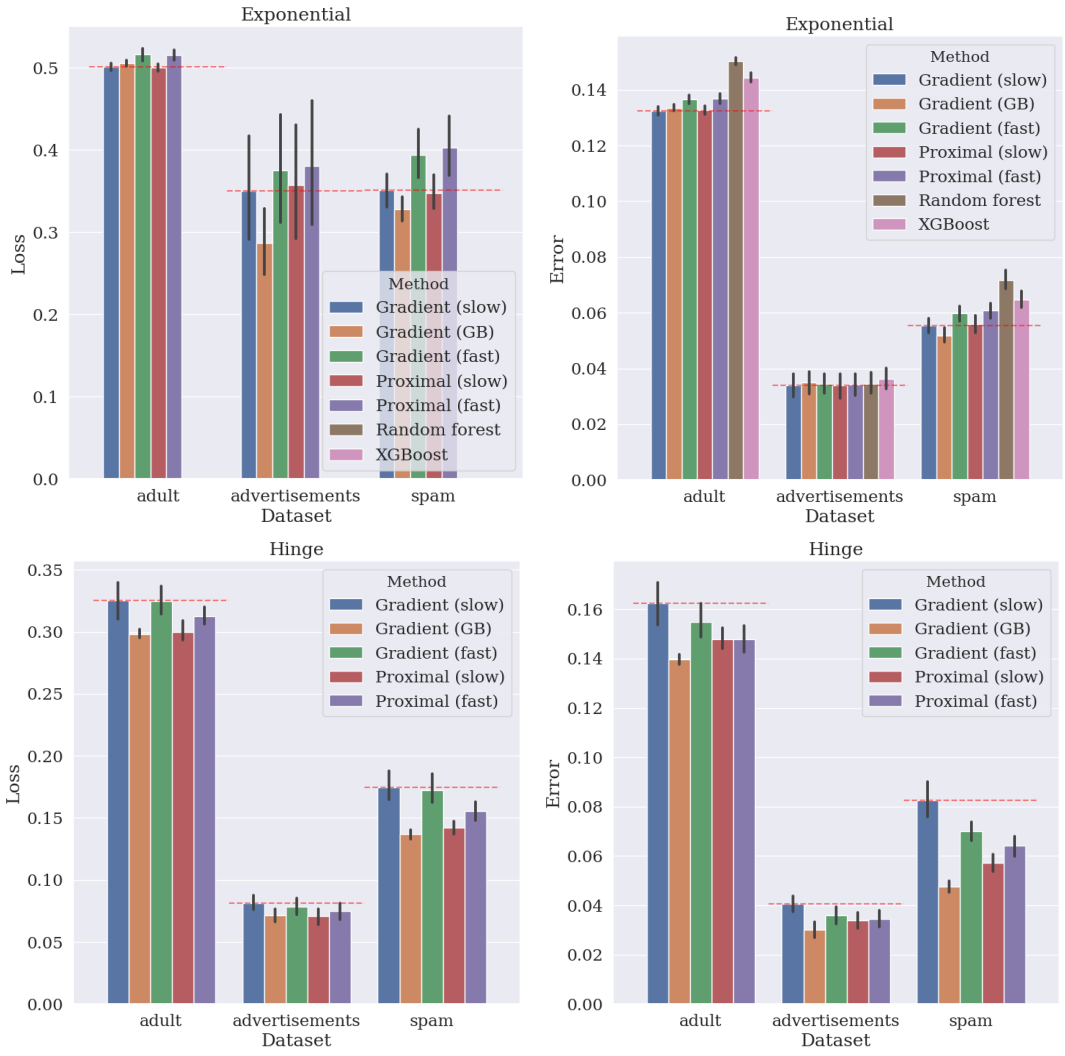

Figure 7: Losses (left) and misclassification rates (right) on the test datasets for the exponential (top) and hinge (bottom) losses.

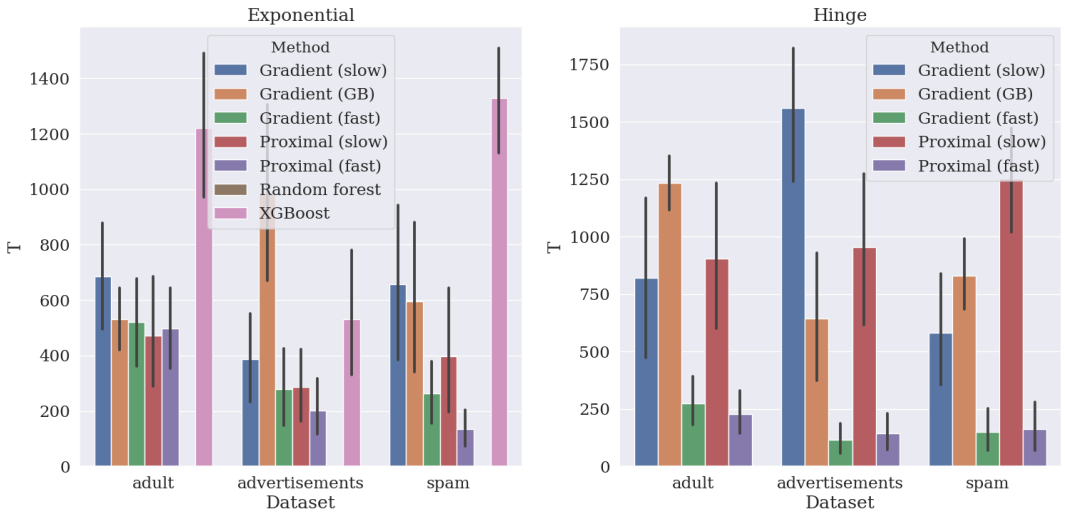

Figure 8: Selected number of weak learners for the exponential (left) and the hinge (right) losses.

6 Conclusion

This paper has introduced a novel boosting algorithm, along with an accelerated variant, which have appeal for non-differentiable loss functions ℓ. The main idea is to use a proximal-based direction of optimization, which can be coupled with Nesterov's acceleration (as already introduced to boosting by Biau et al. [2018]). A theoretical study of the proposed algorithm demonstrates convergence from an optimization point of view and derives recursive formulas to implement efficiently the accelerated variant.

Numerical experiments on synthetic data show a significant impact of the newly introduced parameter λ, but also improvements on regular gradient boosting for adequate values of λ. Moreover, in real-world situations, the proposed proximal boosting algorithm achieves comparable or better accuracies than (extreme) gradient boosting, Grubb and Bagnell's generalized boosting and random forests, depending on the loss employed and the dataset. Overall, proximal boosting tends to use less weak learners than competitors.

Regarding the accelerated variant, accuracy can be a little damaged but final boosting models gain by a dramatically reduced size. Moreover, accelerated proximal boosting may be the most performant model in some situation, such as building a classifier with the hinge loss.

We believe that the connection between boosting and functional optimization can be much more investigated. In particular, advances in optimization theory can spread to boosting, like the recently revisited Frank-Wolfe algorithm impacted boosting [Jaggi, 2013, Wang et al., 2015]. This may also hold true for non-differentiable and non-convex optimization (see for instance [Ochs et al., 2014]).

Acknowledgement

The authors are thankful to Gérard Biau and Jalal Fadili for enlightening discussions.

References

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM journal on imaging sciences, 2(1):183–202, 2009.

G. Biau and B. Cadre. Optimization by gradient boosting. arXiv:1707.05023 [cs, math, stat], 2017.

G. Biau, A. Fischer, B. Guedj, and J.D. Malley. COBRA: A combined regression strategy. Journal of Multivariate Analysis, 146:18–28, 2016.

G. Biau, B. Cadre, and L. Rouvière. Accelerated Gradient Boosting. arXiv:1803.02042 [cs, stat], 2018.

L. Breiman. Arcing the Edge. Technical Report 486, Statistics Department, University of California, Berkeley, 1997.

L. Breiman. Arcing classifier (with discussion and a rejoinder by the author). The Annals of Statistics, 26(3):801–849, 1998.

L. Breiman. Prediction Games and Arcing Algorithms. Neural Computation, 11(7):1493–1517, 1999.

L. Breiman. Some Infinite Theory for Predictor Ensembles. Technical Report 577, Statistics Department, University of California, Berkeley, 2000.

L. Breiman. Random Forests. Machine Learning, 45(1):5–32, 2001.

L. Breiman. Population theory for boosting ensembles. The Annals of Statistics, 32(1):1–11, 2004.

P. Bühlmann and T. Hothorn. Boosting Algorithms: Regularization, Prediction and Model Fitting. Statistical Science, 22(4):477–505, 2007.

T. Chen and C. Guestrin. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 785–794, New York, NY, USA, 2016. ACM.

P. Combettes and V. Wajs. Signal Recovery by Proximal Forward-Backward Splitting. Multiscale Modeling & Simulation, 4(4):1168–1200, 2005.

Y. Freund. Boosting a Weak Learning Algorithm by Majority. Information and Computation, 121(2): 256–285, 1995.

Y. Freund and R.E. Schapire. Experiments with a New Boosting Algorithm. In Proceedings of the Thirteenth International Conference on International Conference on Machine Learning, San Francisco, CA, USA, 1996.

Y. Freund and R.E. Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Journal of Computer and System Sciences, 55(1):119–139, 1997.

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

J. Friedman. Stochastic gradient boosting. Computational Statistics & Data Analysis, 38(4):367–378, February 2002.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). The Annals of Statistics, 28(2):337–407, 2000.

A. Grubb and J.A. Bagnell. Generalized Boosting Algorithms for Convex Optimization. In Proceedings of The 28th International Conference on Machine Learning, Bellevue, Washington, USA, 2011.

M. Jaggi. Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization. In Proceedings of The 30th International Conference on Machine Learning, pages 427–435, Atlanta, GA, USA, 2013.

J. Lin, L. Rosasco, and D.-X. Zhou. Iterative Regularization for Learning with Convex Loss Functions. Journal of Machine Learning Research, 17(77):1–38, 2016.

L. Mason, J. Baxter, P.L. Bartlett, and M. Frean. Boosting Algorithms as Gradient Descent. In S.A. Solla, T.K. Leen, and K. Müller, editors, Advances in Neural Information Processing Systems, pages 512–518. MIT Press, 2000a.

L. Mason, J. Baxter, P.L. Bartlett, and M. Frean. Functional gradient techniques for combining hypotheses. In A.J. Smola, P.L. Bartlett, B. Shölkopf, and D. Schuurmans, editors, Advances in Large Margin Classifiers, pages 221–246. The MIT Press, 2000b.

R. Meir and G. Rätsch. An Introduction to Boosting and Leveraging. In Advanced Lectures on Machine Learning, Lecture Notes in Computer Science, pages 118–183. Springer, Berlin, Heidelberg, 2003.

Y. Nesterov. A method of solving a convex programming problem with convergence rate $0(1/k^2)$. Soviet Mathematics Doklady, 27, 1983.

Y. Nesterov. Introductory Lectures on Convex Optimization: A Basic Course. Kluwer Academic Publishers, 2004.

P. Ochs, Y. Chen, T. Brox, and T. Pock. iPiano: Inertial Proximal Algorithm for Nonconvex Optimization. SIAM Journal on Imaging Sciences, 2014.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

R Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. SIAM journal on control and optimization, 14(5):877–898, 1976.

G. Rätsch, S. Mika, and M.K. Warmuth. On the Convergence of Leveraging. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, Advances in Neural Information Processing Systems, pages 487–494. MIT Press, 2002.

R.E. Schapire. The strength of weak learnability. Machine Learning, 5(2):197–227, 1990.

V.N. Temlyakov. Greedy expansions in convex optimization. In Proceedings of the Steklov Institute of Mathematics, 2012.

C. Wang, Y. Wang, W. E, and R. Schapire. Functional Frank-Wolfe Boosting for General Loss Functions. arXiv:1510.02558 [cs, stat], 2015.

T. Zhang. A General Greedy Approximation Algorithm with Applications. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, Advances in Neural Information Processing Systems 14, pages 1065–1072. MIT Press, 2002.

T. Zhang. Sequential greedy approximation for certain convex optimization problems. IEEE Transactions on Information Theory, 49(3):682–691, March 2003.

A Derivation of boosting weights

When Nesterov's acceleration is on ($(\alpha_t)_{t\in\mathbb{N}}$ is defined by Equation (1)), the update rule becomes (see Line 11 in Algorithm 2):

$$f_{t'+1} = (1+\alpha_{t'})f_{t'} - \alpha_{t'}f_{t'-1} + \nu\gamma_{t'+1}g_{t'+1},$$

for all positive integers $t' \leq T-1$. Let us denote, for each iteration $t' \in \{1, \dots, T-1\}$, $f_{t'} = \sum_{t=0}^{t'} w_t^{(t')} g_t$ the expansion of $f_{t'}$. Then

$$f_{t'+1} = \sum_{t=0}^{t'-1} \left((1+\alpha_{t'}) w_t^{(t')} - \alpha_{t'} w_t^{(t'-1)} \right) g_t + (1+\alpha_{t'}) w_{t'}^{(t')} g_{t'} + \nu \gamma_{t'+1} g_{t'+1}.$$

First, we see that the weights of $g_{t'}$ and $g_{t'+1}$ in the expansion of $f_{t'+1}$ are respectively:

$$\begin{cases} w_{t'}^{(t'+1)} = (1+\alpha_{t'}) w_{t'}^{(t')} \\ w_{t'+1}^{(t'+1)} = \nu \gamma_{t'+1}. \end{cases}$$

Second, for each $t \in \{0, \dots, t'-1\}$, the weight of g_t in the expansion of $f_{t'+1}$ is defined by:

$$w_t^{(t'+1)} = (1+\alpha_{t'}) w_t^{(t')} - \alpha_{t'} w_t^{(t'-1)}.$$

Therefore, considering that weights take value 0 before being defined, i.e. $w_t^{(t-1)} = 0$, we have:

$$w_t^{(t'+1)} - w_t^{(t')} = \alpha_{t'} (w_t^{(t')} - w_t^{(t'-1)}) = \left(\prod_{k=t}^{t'} \alpha_k \right) (w_t^{(t)} - w_t^{(t-1)}) = \left(\prod_{k=t}^{t'} \alpha_k \right) w_t^{(t)}.$$

It follows that:

$$w_t^{(t'+1)} = w_t^{(t')} + \left(\prod_{k=t}^{t'} \alpha_k \right) w_t^{(t)} = w_t^{(t)} + \sum_{j=t}^{t'} \left(\prod_{k=t}^{j} \alpha_k \right) w_t^{(t)} = \left(1 + \sum_{j=t}^{t'} \prod_{k=t}^{j} \alpha_k \right) w_t^{(t)}.$$

Then, for $k \leq 1$, one has $\alpha_k = 0$, so $w_0^{(t'+1)} = w_0^{(0)} = 1$ and $w_1^{(t'+1)} = w_1^{(1)} = \nu \gamma_1$. Now, remarking that, for all $t \geq 2$, $w_t^{(t)} = \nu \gamma_t$, we can conclude that the weights of f_T are:

$$\begin{cases} w_0 = 1 \\ w_1 = \nu \gamma_1 \\ w_t = \left(1 + \sum_{j=t}^{T-1} \prod_{k=t}^{j} \alpha_k \right) \nu \gamma_t, \forall t \in \{2, \dots, T-1\} \\ w_T = \nu \gamma_T. \end{cases}$$

In a computational perspective, it may be efficient to update the weights, at each iteration t, according to the following recursion:

$$\begin{cases} w_0^{(0)} = 1 \\ w_1^{(0)} = \nu \gamma_1 \\ w_1^{(1)} = \nu \gamma_1 \\ w_j^{(t+1)} = (w_j^{(t)} - w_j^{(t-1)})(1+\alpha_t) + w_j^{(t-1)}, \forall j \in \{1, \dots, t\} \\ w_{t+1}^{(t+1)} = \nu \gamma_{t+1}. \end{cases} \tag{8}$$

B Analysis of the approximated proximal point method

Let us consider the optimization problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ F(x), \tag{P2}$$

where $F : \mathbb{R}^n \to \mathbb{R}$ is:

(C) convex and differentiable;

(SM) L-strongly smooth (for some $L > 0$): $\forall x, x' \in \mathbb{R}^n$:

$$F(x') \leq F(x) + \langle \nabla F(x), x' - x \rangle_{\ell_2} + \frac{L}{2} \|x' - x\|_{\ell_2}^2 ;$$

(SC) κ-strongly convex (for some $\kappa > 0$): $\forall x, x' \in \mathbb{R}^n$:

$$F(x') \geq F(x) + \langle \nabla F(x), x' - x \rangle_{\ell_2} + \frac{\kappa}{2} \|x' - x\|_{\ell_2}^2 .$$

For an operator $P : \mathbb{R}^n \to \mathbb{R}^n$, we consider the approximated proximal point method, described in Algorithm 3. It is similar to the proximal point iteration but makes use of a modified direction of update ($P(g_t)$ instead of g_t). In particular, let us remark that when $P(x) = x$, Algorithm 3 recovers the original proximal point method.

Algorithm 3 Approximated proximal point method.

Input: $\lambda > 0$ (proximal step), $P : \mathbb{R}^n \to \mathbb{R}^n$ (approximation operator).

1: Set $x_0 \in \mathbb{R}^n$ (initialization).
2: **for** $t = 0$ **to** $T - 1$ **do**
3: $\quad g_t \leftarrow \frac{1}{\lambda} (x_t - \operatorname{prox}_{\lambda F}(x_t))$.
4: $\quad$ Set $x_{t+1} = x_t - \lambda P(g_t)$.
5: **end for**

Output: x_T.

Linear convergence of Algorithm 3 under the edge property (Assumption (9)) is stated in Theorem 5.

Theorem 5. *Let $F : \mathbb{R}^n \to \mathbb{R}$ be a function satisfying Assumptions (C), (SM) and (SC) and $P : \mathbb{R}^n \to \mathbb{R}^n$ be any operator. Let $(x_t)_t$ be a sequence generated by Algorithm 3 and assume that there exists $\zeta \in (0, 1]$ such that for any iteration t:*

$$\|g_t - P(g_t)\|_{\ell_2}^2 \leq (1 - \zeta^2) \|g_t\|_{\ell_2}^2 . \tag{9}$$

Let $\{x^\star\} \in \arg\min_{x \in \mathbb{R}^n} F(x)$ (well defined by strong convexity), and choose $\lambda = \frac{\zeta^2}{8L}$. Then,

$$F(x_T) - F(x^\star) \leq \left(1 - \frac{\zeta^2 \kappa}{9L}\right)^T (F(x_0) - F(x^\star)) .$$

Proof. First of all, let us remark that:

1. Assumptions (C) and (SM) imply L-Lipschitz continuity of ∇F [Nesterov, 2004, Theorem 2.1.5]:

$$\forall x, x' \in \mathbb{R}^n : \qquad \|\nabla F(x) - \nabla F(x')\|_{\ell_2} \leq L \|x - x'\|_{\ell_2} ; \tag{10}$$

2. Assumptions (C) and (SC) lead to the upper bound [Nesterov, 2004, Theorem 2.1.10]:

$$\forall x \in \mathbb{R}^n : \qquad 2\kappa (F(x) - F(x^\star)) \leq \|\nabla F(x)\|_{\ell_2} . \tag{11}$$

From Assumption (SM) and by the update rule for x_{t+1} in Algorithm 3:

$$\begin{aligned} F(x_{t+1}) &\leq F(x_t) + \langle \nabla F(x_t), -\lambda P(g_t) \rangle + \frac{L\lambda^2}{2} \|P(g_t)\|_{\ell_2}^2 \\ &\leq F(x_t) - \lambda \langle g_t, P(g_t) \rangle - \lambda \langle \nabla F(x_t) - g_t, P(g_t) \rangle + \frac{L\lambda^2}{2} \|P(g_t)\|_{\ell_2}^2 . \end{aligned} \tag{12}$$

Now, from Assumption (9):

$$\begin{aligned} -\lambda \langle g_t, P(g_t) \rangle &= \frac{\lambda}{2} \left(\|g_t - P(g_t)\|_{\ell_2}^2 - \|g_t\|_{\ell_2}^2 - \|P(g_t)\|_{\ell_2}^2 \right) \\ &\leq \frac{\lambda}{2} \left[(1 - \zeta^2) \|g_t\|_{\ell_2}^2 - \|g_t\|_{\ell_2}^2 - \|P(g_t)\|_{\ell_2}^2 \right] \\ &= -\frac{\lambda \zeta^2}{2} \|g_t\|_{\ell_2}^2 - \frac{\lambda}{2} \|P(g_t)\|_{\ell_2}^2 . \end{aligned} \tag{13}$$

Besides, given that, by definition of the proximal operator, $g_t = \nabla F(\text{prox}_{\lambda F}(x_t))$, one has:

$$\begin{aligned} \|\nabla F(x_t) - g_t\|_{\ell_2} &= \|\nabla F(x_t) - \nabla F\left(\text{prox}_{\lambda F}(x_t)\right)\|_{\ell_2} \\ &\leq L \|x_t - \text{prox}_{\lambda F}(x_t)\|_{\ell_2} && \text{(using Equation (10))} \\ &\leq \lambda L \|g_t\|_{\ell_2} && \text{(definition of } g_t\text{).} \end{aligned} \tag{14}$$

So,

$$\begin{aligned} -\lambda \langle \nabla F(x_t) - g_t, P(g_t) \rangle &\leq \lambda \|\nabla F(x_t) - g_t\|_{\ell_2} \|P(g_t)\|_{\ell_2} && \text{(Cauchy-Schwarz)} \\ &\leq \lambda^2 L \|g_t\|_{\ell_2} \|P(g_t)\|_{\ell_2} && \text{(Equation (14))} \\ &\leq 2\lambda^2 L \|g_t\|_{\ell_2}^2 && (\|P(g_t)\|_{\ell_2} \leq 2 \|g_t\|_{\ell_2} \text{ by Assumption (9)).} \end{aligned} \tag{15}$$

Combining Equations (12), (13) and (15):

$$\begin{aligned} F(x_{t+1}) &\leq F(x_t) - \frac{\lambda \zeta^2}{2} \|g_t\|_{\ell_2}^2 - \frac{\lambda}{2} \|P(g_t)\|_{\ell_2}^2 + 2\lambda^2 L \|g_t\|_{\ell_2}^2 + \frac{L\lambda^2}{2} \|P(g_t)\|_{\ell_2}^2 \\ &= F(x_t) - \lambda \left(\frac{\zeta^2}{2} - 2\lambda L \right) \|g_t\|_{\ell_2}^2 - \frac{\lambda}{2} (1 - L\lambda) \|P(g_t)\|_{\ell_2}^2 . \end{aligned}$$

Now, choosing $\lambda = \frac{\zeta^2}{8L}$, one has $-\frac{\lambda}{2}(1 - L\lambda) \|P(g_t)\|_{\ell_2}^2 \leq 0$ and $\left(\frac{\zeta^2}{2} - 2\lambda L \right) = \frac{1}{2}$, leading to:

$$F(x_{t+1}) \leq F(x_t) - \frac{\zeta^4}{16L} \|g_t\|_{\ell_2}^2 . \tag{16}$$

Let us remark that, by Equation (11):

$$\begin{aligned} 2\kappa \left(F(x_t) - F(x^\star) \right) &\leq \|\nabla F(x_t)\|_{\ell_2} \\ &\leq \|\nabla F(x_t) - g_t\|_{\ell_2} + \|g_t\|_{\ell_2} \\ &\leq (1 + \lambda L) \|g_t\|_{\ell_2} && \text{(Equation (14))} \\ &\leq \left(1 + \frac{\zeta^2}{8} \right) \|g_t\|_{\ell_2} && \left(\lambda = \frac{\zeta^2}{8L} \right), \end{aligned}$$

that is,

$$\|g_t\|_{\ell_2} \geq \frac{16\kappa}{8 + \zeta^2} \left(F(x_t) - F(x^\star) \right) .$$

So, from Equation (16),

$$\begin{aligned}
F(x_{t+1}) - F(x^\star) &\leq F(x_t) - F(x^\star) - \frac{\zeta^4}{16L} \|g_t\|_{\ell_2}^2 \\
&\leq \left(1 - \frac{\zeta^4}{16L} \frac{16\kappa}{8+\zeta^2}\right) (F(x_t) - F(x^\star)) \\
&= \left(1 - \frac{\kappa}{L} \frac{\zeta^4}{8+\zeta^2}\right) (F(x_t) - F(x^\star)) \\
&\leq \left(1 - \frac{\kappa}{L} \frac{\zeta^2}{9}\right) (F(x_t) - F(x^\star)) \qquad \left(\forall x \in [0,1], \quad \frac{x^2}{8+x} \geq \frac{x^2}{9}\right) \\
&\leq \left(1 - \frac{\zeta^2 \kappa}{9L}\right)^{t+1} (F(x_0) - F(x^\star)) \qquad \text{(by induction).}
\end{aligned}$$

□

C Additional numerical results

C.1 Uncorrelated synthetic dataset

This section provides the numerical results for a variant of the regression and classification models used in Section 5.

Regression : $n = 800, d = 100, Y = -\sin(2X_1) + X_2^2 + X_3 - \exp(-X_4) + Z_{0,0.5}$.

Classification : $n = 1500, d = 50, Y = \mathbf{21}_{X_1 + X_4^3 + X_9 + \sin(X_{12} X_{18}) + Z_{0,0.1} > 0.38} - 1$.

In both cases, we consider an input random variable $X \in \mathbb{R}^d$, the covariate of which, denoted $(x_j)_{1 \leq j \leq d}$, are uniformly distributed over $(-1,1)^d$ (uncorrelated design) and the random variable Z_{μ,σ^2} represents an additive and independent noise normally distributed with mean $\mu \in \mathbb{R}$ and variance σ^2.

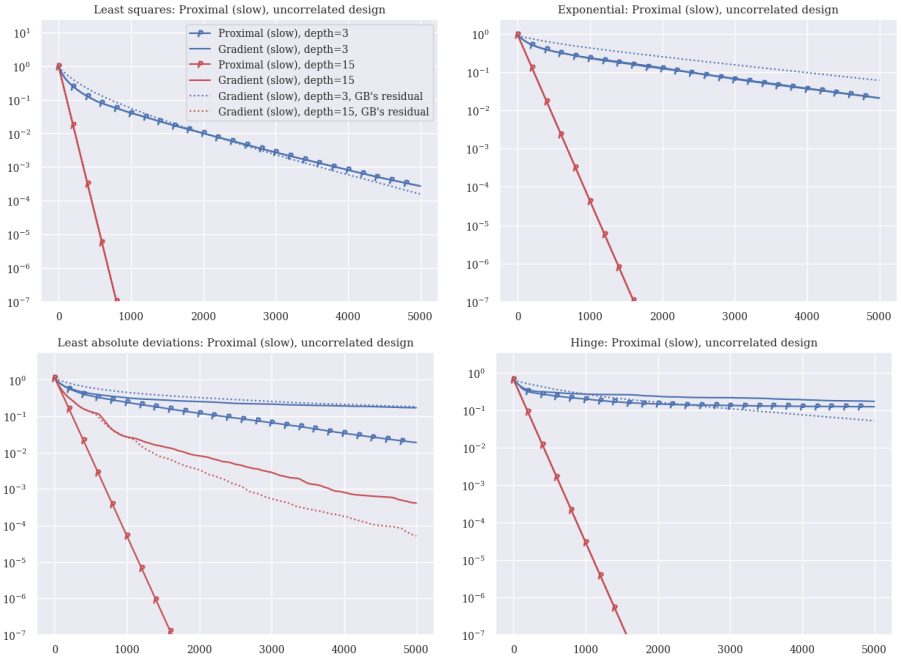

Figure 9: Training losses for two values of maximal depth (3 in blue, 15 in red) and Grubb and Bagnell's residual (dashed lines) – number of iterations on the horizontal axis.

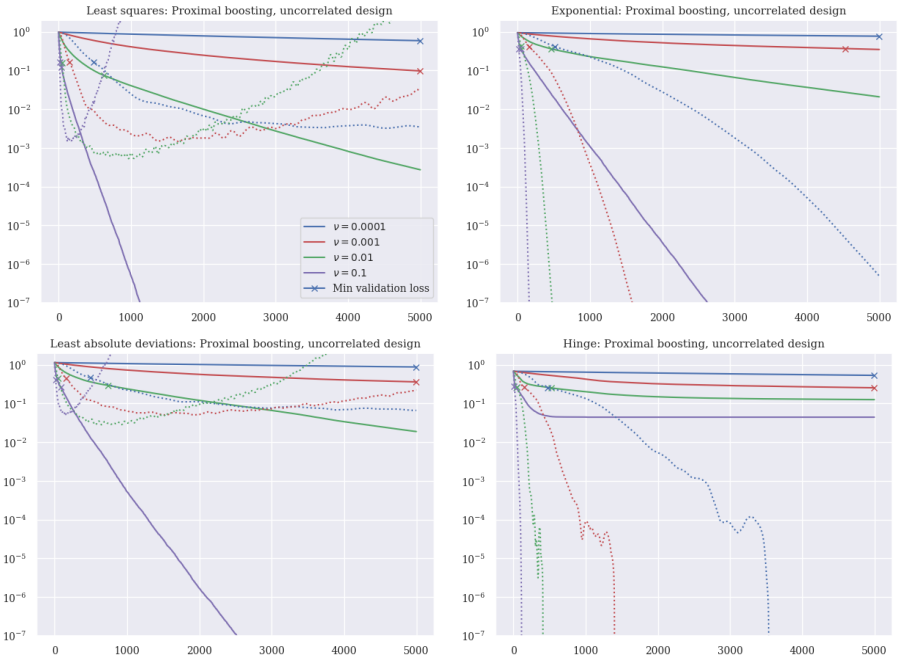

Figure 10: Training losses of proximal boosting (solid line) and its accelerated version (dashed line) for several values of ν – number of iterations on the horizontal axis. For each curve, the iteration where the minimum validation loss is achieved is represented by a cross.

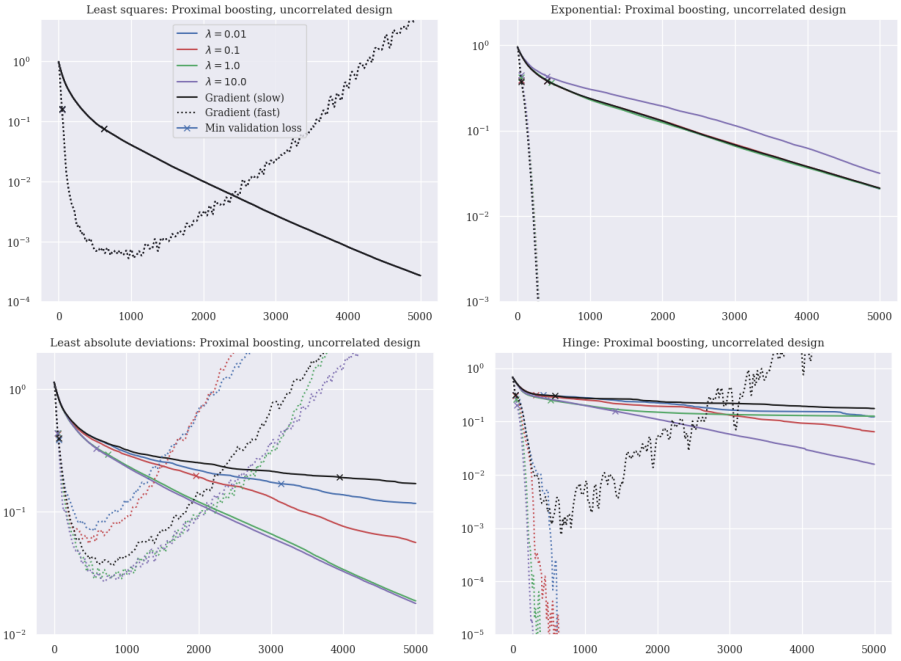

Figure 11: Training losses of proximal boosting (solid line) and its accelerated version (dashed line) for several values of λ – number of iterations on the horizontal axis. For each curve, the iteration where the minimum validation loss is achieved is represented by a cross.

			Slow					Fast			
Loss	Method	Depth	T	ν	λ	Error	Depth	T	ν	λ	Error
Least squares	Gradient	1	503	8e-02	-	**0.31** (0.02)	1	405	2e-02	-	**0.31** (0.02)
	Gradient (GB)	3	2399	1e-02	-	0.33 (0.03)					
	Proximal	1	503	8e-02	5e-01	**0.31** (0.02)	1	405	2e-02	1e-01	**0.31** (0.02)
Least absolute deviations	Gradient	1	1501	8e-02	-	0.65 (0.03)	1	179	5e-02	-	0.64 (0.02)
	Gradient (GB)	2	3011	8e-02	-	0.69 (0.03)					
	Proximal	1	510	8e-02	6	**0.63** (0.02)	1	284	2e-02	7	**0.63** (0.02)
Exponential	Gradient	1	237	1e-01	-	**0.54** (0.04)	1	346	2e-02	-	0.55 (0.04)
	Gradient (GB)	3	1972	2e-02	-	0.56 (0.04)					
	Proximal	1	233	1e-01	7e-01	**0.54** (0.04)	1	425	4e-02	9e-01	**0.54** (0.04)
Hinge	Gradient	3	681	7e-02	-	0.39 (0.04)	1	433	3e-02	-	0.37 (0.03)
	Gradient (GB)	3	95	1e-01	-	0.41 (0.03)					
	Proximal	2	1362	5e-02	5e-02	0.37 (0.03)	2	305	3e-02	2	**0.35** (0.03)

Table 4: Generalization ability: average minimal validation errors (with standard deviations) and associated parameters. For each block, boldface values are smallest errors.

C.2 Real-world datasets

This part completes Section 5.2 by providing the average value of the maximal depth selected for base learners.

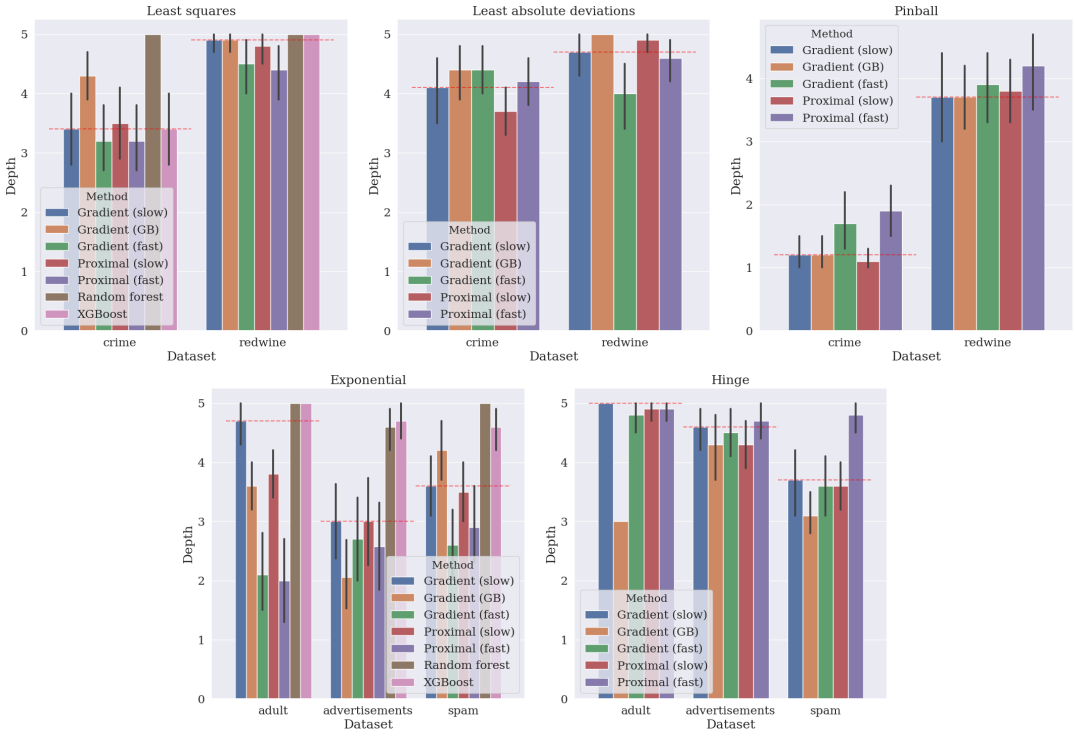

Figure 12: Selected value of maximal depth for the least squares (top left), least absolute deviations (top middle) and pinball (top right), exponential (bottom left) and hinge (bottom right) losses.

D Implementation details

As explained previously, given a loss function $\ell\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, gradient and proximal boosting aim at minimizing the risk functional

$$C(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, F(X_i)) = D(f(X_1^n))$$

for $f \in \operatorname{span} \mathcal{F}$ (where $\mathcal{F}$ is a class of weak learners $f\colon \mathbb{R}^d \to \mathbb{R}$), thereby measuring the cost incurred by predicting $f(X_i)$ when the answer is Y_i.

In the forthcoming subsections, implementation details are given for six popular losses: least squares, least absolute deviations and pinball losses (regression), as well as exponential, logistic and hinge losses (binary classification). In this latter case, the predicted label of a point $x \in \mathbb{R}^d$ is given by $+1$ if $f(x) \geq 0$ and -1 otherwise.

For each loss, we lay out the following information:

Definition: the mapping of the loss function $\ell\colon (y, y') \in \mathbb{R}^2 \mapsto \ell(y, y')$.

Initial estimator: the constant function $f_0 \in \arg\min_{f \in \mathcal{F}_0} C(f)$.

Line search: the optimal step size $\gamma_{t+1} \in \arg\min_{\gamma \in \mathbb{R}} C(f_t + \gamma g_t)$.

(Sub)gradient: the direction of optimization to follow at the iterate f_t.

Proximal operator: for all $z \in \mathbb{R}^n$, $\operatorname{prox}_{\lambda n D}(z) = \arg\min_{u \in \mathbb{R}^d} \lambda n D(u) + \frac{1}{2} \|u - z\|_{\ell_2}^2$.

First, for the exponential and the logistic loss, the line search and the proximal operator have no closed-form solution, but are known to be roots of some equations. In that case, we perform one or several steps of the Newton-Raphson method to obtain an approximation of the desired quantity.

Second, when using decision trees as base learners, its common to perform a line search for each leaf of the tree g_t. In that case, the line search may take a simpler form than the one given below.

D.1 Least squares

Definition: $\ell(y, y') = (y - y')^2/2$.

Initial estimator: $f_0 = \frac{1}{n}\sum_{i=1}^n Y_i$.

Line search: $\gamma_{t+1} = \begin{cases} \frac{\sum_{i=1}^n (Y_i - f_t(X_i)) g_{t+1}(X_i)}{\sum_{i=1}^n g_{t+1}(X_i)^2} & \text{if } \sum_{i=1}^n g_{t+1}(X_i)^2 > 0 \\ 0 & \text{otherwise.} \end{cases}$

Gradient: $\nabla_{\mathrm{n}} C(f_t) = ((f_t(X_i) - Y_i)/n)_{1 \leq i \leq n}$.

Proximal operator: $\mathrm{prox}_{\lambda n D}(z) = ((\lambda Y_i + z_i)/(1 + \lambda))_{1 \leq i \leq n}$.

D.2 Least absolute deviations

Definition: $\ell(y, y') = |y - y'|$.

Initial estimator: f_0 is the empirical median of the sample $\{y_1, \ldots, y_n\}$.

Line search: $\gamma_{t+1} = \arg\min_{\gamma \in \{0\} \cup \left\{ \frac{Y_i - f_t(X_i)}{g_{t+1}(X_i)} : g_{t+1}(X_i) \neq 0 \right\}} C(f_t + \gamma g_t)$.

Subradient: $\widetilde{\nabla}_{\mathrm{n}} C(f_t) = ((\mathrm{sign}(f_t(X_i) - Y_i))/n)_{1 \leq i \leq n}$, where for all $x \in \mathbb{R}$, $\mathrm{sign}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases}$

Proximal operator: $\mathrm{prox}_{\lambda n D}(z) = \left(\max\left(0, 1 - \frac{\lambda}{|z_i - Y_i|}\right)(z_i - Y_i) + Y_i \right)_{1 \leq i \leq n}$.

D.3 Pinball

Definition: $\ell(y, y') = \max(\tau(y - y'), (\tau - 1)(y - y'))$, $\tau \in (0, 1)$.

Initial estimator: f_0 is the τ-quantile of the sample $\{y_1, \ldots, y_n\}$.

Line search: $\gamma_{t+1} = \arg\min_{\gamma \in \{0\} \cup \left\{ \frac{Y_i - f_t(X_i)}{g_{t+1}(X_i)} : g_{t+1}(X_i) \neq 0 \right\}} C(f_t + \gamma g_t)$.

Subradient: $\widetilde{\nabla}_{\mathrm{n}} C(f_t) = \left(\begin{cases} (\tau - 1)/n & \text{if } Y_i - f_t(X_i) < 0 \\ \tau/n & \text{if } Y_i - f_t(X_i) > 0 \\ 0 & \text{otherwise} \end{cases} \right)_{1 \leq i \leq n}$.

Proximal operator: $\mathrm{prox}_{\lambda n D}(z) = \left(\begin{cases} z_i + \lambda\tau & \text{if } Y_i - z_i > \lambda\tau \\ z_i + \lambda(\tau - 1) & \text{if } Y_i - z_i < \lambda(\tau - 1) \\ Y_i & \text{otherwise} \end{cases} \right)_{1 \leq i \leq n}$.

D.4 Exponential loss

Definition: $\ell(y, y') = \exp(-\beta y y')$, $\beta > 0$.

Initial estimator: $f_0 = \frac{\log\left(\frac{p}{n-p}\right)}{2\beta}$, where $p = \sum_{\substack{1 \leq i \leq n \\ Y_i = 1}} 1$.

Line search: no closed-form solution.

Gradient: $\nabla_{\mathrm{n}} C(f_t) = \left(\frac{-\beta Y_i e^{-Y_i f_t(X_i)}}{n}\right)_{1 \leq i \leq n}$.

Proximal operator: no closed-form solution.

D.5 Logistic loss

Definition: $\ell(y, y') = \log_2(1 + \exp(-y y'))$.

Initial estimator: $f_0 = \log\left(\frac{p}{n-p}\right)$, where $p = \sum_{\substack{1 \leq i \leq n \\ Y_i = 1}} 1$.

Line search: no closed-form solution.

Gradient: $\nabla_{\mathrm{n}} C(f_t) = \left(\frac{-Y_i e^{-Y_i f_t(X_i)}}{n \log_2(1 + e^{-Y_i f_t(X_i)})}\right)_{1 \leq i \leq n}$.

Proximal operator: no closed-form solution.

D.6 Hinge loss

Definition: $\ell(y, y') = \max(0, 1 - y y')$.

Initial estimator: $f_0 = \operatorname{sign}\left(\sum_{i=1}^n Y_i\right)$.

Line search: $\gamma_{t+1} = \arg\min_{\gamma \in \{0\} \cup \left\{\frac{1 - Y_i f_t(X_i)}{Y_i g_{t+1}(X_i)} : g_{t+1}(X_i) \neq 0\right\}} C(f_t + \gamma g_t)$.

Subgradient: $\widetilde{\nabla}_{\mathrm{n}} C(f_t) = \left(\begin{cases} -Y_i/n & \text{if } Y_i f_t(X_i) < 1 \\ 0 & \text{otherwise} \end{cases}\right)_{1 \leq i \leq n}$.

Proximal operator: $\operatorname{prox}_{\lambda n D}(z) = \left(\begin{cases} z_i + \lambda Y_i & \text{if } Y_i z_i < 1 - \lambda \\ z_i & \text{if } Y_i z_i > 1 \\ Y_i & \text{otherwise} \end{cases}\right)_{1 \leq i \leq n}$.