Proximal boosting: aggregating weak learners to minimize non-differentiable losses

Erwan Fouillen, Claire Boyer, and Maxime Sangnier

Université de Paris and Sorbonne Université, CNRS, Laboratoire de Probabilités, Statistique et Modélisation, F-75013 Paris, France

November 29, 2022

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, in order to introduce a novel boosting approach, called proximal boosting. It comes with a companion algorithm inspired by [Grubb and Bagnell, 2011] and called residual proximal boosting, which is aimed at better controlling the approximation error. Theoretical convergence is proved for these two procedures under different hypotheses on the empirical risk and 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 sequentially 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. It is thus possible to reduce the bias of the risk while controlling its variance. This is a noticeable advantage over very complex models such as nonparametric methods. ii) from the data science perspective, fitting a boosting model is computationally cheap, making it possible to be used on large datasets. In contrast, it can quickly achieve sufficiently complex models to be 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 for tabular data in several real-world situations ranging from data challenges to tangible applications in urbanization [Ikeagwuani et al., 2021, Rajendran et al., 2021], renewable energy [Tyralis and Papacharalampous, 2021, Chen et al., 2022] and medical care [Ahamad et al., 2020, Awal et al., 2021, Santana et al., 2021].

As explained by Biau et al. [2019], gradient boosting has its roots in Freund and Schapire's work on combining classifiers, which resulted in the Adaboost algorithm [Freund and Schapire, 1997, Schapire,

1990, Freund, 1995, Freund and Schapire, 1996]. Later, Friedman and colleagues developed a novel boosting procedure inspired by the numerical optimization literature, and nicknamed gradient boosting [Friedman, 2001, Friedman et al., 2000, Friedman, 2002]. Such a connection of boosting to 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, 2014, Biau and Cadre, 2021]. Let us remark that rates of convergence of gradient boosting are known for smooth and strongly convex risks [Grubb and Bagnell, 2011, Rätsch et al., 2002].

Since the invention of gradient boosting, several variants have been emerging (see for instance [Bühlmann and Yu, 2003, Zhang and Yu, 2005, Gao and Koller, 2011, Wang et al., 2019] to cite only a few) up to very recent studies concerning large-scale regression with boosted histograms [Cai et al., 2020, Cui et al., 2021, Hang et al., 2021]. The statistical properties of boosting algorithms have been addressed many times (for instance in [Bühlmann and Yu, 2003, Park et al., 2009, Lin et al., 2019]) and are still under consideration as a modern topic of statistical learning [Cai et al., 2020, Cui et al., 2021, Hang et al., 2021, Zeng et al., 2022].

In practice, the number of weak learners used in gradient boosting (and variants) controls the statistical complexity of the final predictor but also the number of optimization steps performed in order to minimize the empirical risk. While controlling the latter is a natural way to regularize the method and to enhance its generalization properties, tuning the former makes it possible to stop the optimization algorithm before convergence, which is known in many areas as early stopping. This technique can be seen as an iterative regularization mechanism also used to prevent overfitting [Lin et al., 2016]. As a consequence, besides its approximation capability, the statistical performance of gradient boosting deeply relies on the algorithm employed.

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., 2019, Lu et al., 2020]. While all these variants rely on differentiable loss functions, Grubb and Bagnell [2011] discusses 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, consisting in reprojecting the error made when approximating the subgradients by weak learners. The contribution of the work described here is to go a step forward by proposing novel procedures to efficiently learn boosted models with non-differentiable loss functions.

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. Building upon a background on non-smooth optimization, Section 3 encloses the main contribution of this paper: adapting the proximal point algorithm [Nesterov, 2004] to boosting. The proposed method is nicknamed proximal boosting and comes with a variant, called residual proximal boosting, inspired by Grubb and Bagnell [2011]. A second contribution is to prove convergence rates (from an optimization perspective) of proximal and residual proximal boosting under different hypotheses on the loss function (see Section 4). Finally, the numerical study described in Section 5 shines a light on advantages and limitations of the proposed boosting procedures. As a by-product, we also consider adapting Nesterov's acceleration to proximal boosting, such as in accelerated gradient boosting [Biau et al., 2019]. Even though our proposed algorithm performs better than that of [Biau et al., 2019], we observe divergence on the training set (as this is the case for accelerated gradient boosting [Biau et al., 2019, Lu et al., 2020]) and no particular gain in accuracy.

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 measurable 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 a minimizer of the risk $\mathbb{E}(\ell(Y, f(X)))$ over all measurable 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), $\mathscr{F}_{\mathcal{X}}$ the set of functions from $\mathcal{X}$ to $\mathbb{R}$ and $\mathcal{F} \subseteq \mathscr{F}_{\mathcal{X}}$ a class of functions. 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

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

is the empirical risk and

$$\operatorname{span} \mathcal{F} = \left\{ \sum_{t=1}^{m} w_t g_t : w \in \mathbb{R}^m, (g_1, \ldots, g_m) \in \mathcal{F}^m, m \in \mathbb{N} \right\}$$

is the set of all linear combinations of functions in $\mathcal{F}$ ($\mathbb{N}$ being the set of non-negative integers).

As a simple example, let us consider the regression model $Y = \sin(2\pi X) + \sin(32\pi X) + \epsilon$, where X is uniformly distributed on $[0, 1]$ and ϵ is normally distributed and independent of X. We aim at solving:

$$\underset{f \in \operatorname{span} \mathcal{F}}{\text{minimize}} \; \frac{1}{n} \sum_{i=1}^{n} |Y_i - f(X_i)|,$$

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 500): a traditional one with a subgradient-type method (Algorithm 1, Section 3.2), and another with the proposed proximal-based procedure (Algorithm 2, Section 3.3). Figure 1 depicts the prediction of f_T (left) and the training error $C(f_t) = \frac{1}{n} \sum_{i=1}^{n} |Y_i - f_t(X_i)|$ along the iterations t (right, green curve).

From 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 attains a plateau at $1.2 \cdot 10^{-1}$) 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 prevail over subgradient techniques for non-differentiable objective functions.

Beyond optimization, proximal boosting also outperforms gradient boosting from a statistical perspective since it achieves a lower test error (red curve in the right side of Figure 1).

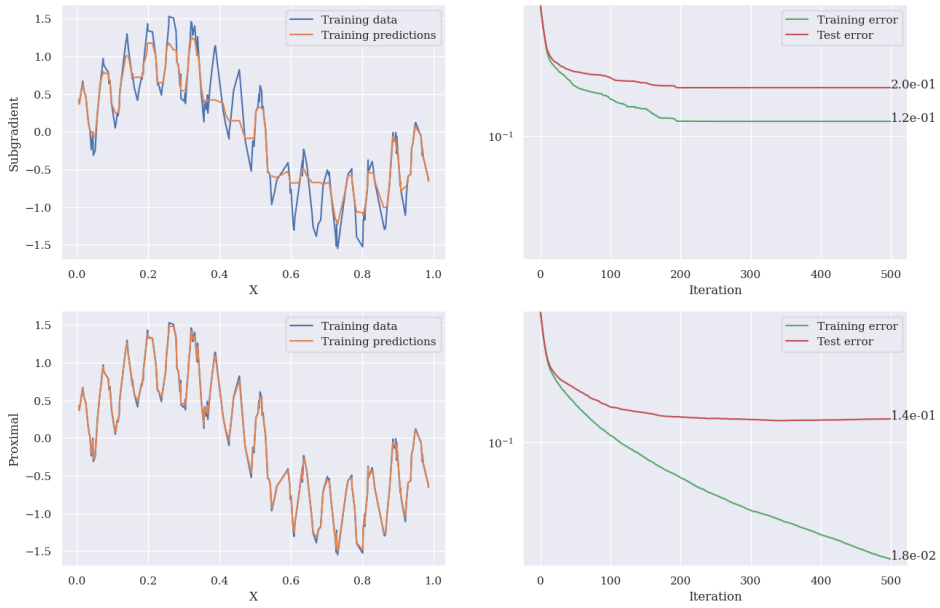

Figure 1: Predicted values and training error of a boosting machine trained with a subgradient (top) and a proximal-based method (bottom).

3 Algorithms

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 the vector $f(X_1^n) = (f(X_1), \dots, f(X_n)) \in \mathbb{R}^n$. 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, for any $f \in \mathscr{F}_{\mathcal{X}}$, $C(f) = D(f(X_1^n))$.

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 match the vectors appearing in standard optimization procedures with functions from $\mathscr{F}_{\mathcal{X}}$. In particular, given a vectorial gradient $\nabla D(f(X_1^n))$ ($f \in \mathscr{F}_{\mathcal{X}}$), one has to find a function $g \in \mathscr{F}_{\mathcal{X}}$ 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 $f \in \mathscr{F}_{\mathcal{X}}$, the functional gradient of C as $\nabla_n C(f) = \nabla D(f(X_1^n))$. On the contrary, if ℓ is not differentiable, we may consider a subgradient of C at f, denoted $\widetilde{\nabla}_n C(f)$ 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. For the sake of readability, the algorithms introduced in the next sections are recapped in Table 1.

3.1 The 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

Algorithm	Description
1	Gradient boosting with convergence rate $O(1/t)$ for smooth losses
2	Proximal boosting with convergence rate $O(1/t)$ for smooth losses
3	Proximal boosting with convergence rate $O(1/\sqrt{t})$ for non-smooth losses
4	Proximal boosting with Nesterov's acceleration
5	Numerical implementation of Algorithms 1, 2 and 3 with shrunk step size
6	Abstract algorithm for the proof of Theorem 2
7	Abstract algorithm for the proof of Theorem 3
8	Numerical implementation of Algorithm 4 with shrunk step size

Table 1: Summary of algorithms.

and lower semi-continuous. Besides, let us define the proximal operator of h by:

$$\operatorname{prox}_h(x) = \arg\min_{u \in \mathbb{R}^d} \left\{ h(u) + \frac{1}{2} \|u - x\|_2^2 \right\}, \qquad \forall x \in \mathbb{R}^d,$$

where $\|\cdot\|_2$ is the Euclidean norm. This operator is well defined by convexity and lower semi-continuity of h [Combettes and Wajs, 2005]. Then, the iterative procedure defined by choosing any $x_0 \in \mathbb{R}^d$ and by setting for all $t \in \mathbb{N}$:

$$x_{t+1} = \operatorname{prox}_{\gamma_{t+1} h}(x_t - \gamma_{t+1} \nabla g(x_t)),$$

where $\gamma_{t+1} \in (0, 2/L)$, is known as the proximal gradient method, and converges to a minimizer of $g + h$ in $O(1/t)$ [Nesterov, 2004]. More formally, assuming that $g + h$ has a minimizer $x^\star$, then $(g + h)(x_t) - (g + h)(x^\star) = O(1/t)$.

Depending on the properties of the objective function to minimize, the procedure described before leads to two simple algorithms:

- the gradient method ($h = 0$):

  $$x_{t+1} = x_t - \gamma_{t+1} \nabla g(x_t),$$

  minimizes a single function g as soon as it is convex and differentiable with Lipschitz continuous gradient;

- the proximal point algorithm ($g = 0$):

  $$x_{t+1} = \operatorname{prox}_{\gamma_{t+1} h}(x_t) = x_t - \gamma_{t+1} \left[\frac{1}{\gamma_{t+1}} \left(x_t - \operatorname{prox}_{\gamma_{t+1} h}(x_t) \right) \right], \tag{1}$$

  minimizes a single function h, which is only required to be convex and lower semi-continuous (in this case, there is no restriction on the step size γ_{t+1}, except being positive).

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 $O(1/t)$. 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, with the iteration given in Equation (1).

This has to be put in contrast with the subgradient method:

$$x_{t+1} = x_t - \gamma_{t+1} \widetilde{\nabla} h(x_t), \tag{2}$$

where $\gamma_{t+1} > 0$ 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 [Nesterov, 2004]. In addition, this rate is tight for this optimization procedure: it cannot be improved without extra assumptions on h [Nesterov, 2004, Theorem 3.2.1].

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 (1)). This motivation is emphasized by the fact that moving from the subgradient to the proximal point method only requires to replace the update direction $\widetilde{\nabla} h(x_t)$ by $\frac{1}{\gamma_{t+1}}(x_t - \mathrm{prox}_{\gamma_{t+1} h}(x_t))$. This observation is the cornerstone of the boosting algorithms 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 $\mathcal{F}_0 \subseteq \mathcal{F}$. Then, a simple procedure to approximately solve (P1) is gradient boosting, described in Algorithm 1 [Friedman, 2001, Mason et al., 2000a]. It builds the requested additive model in an iterative fashion, by imitating a gradient method (or subgradient method if ℓ is not differentiable with respect to its second argument). 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 positive weight $w_{t+1} = \gamma_{t+1}$. 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 Gradient boosting.

Input: $\gamma_1, \ldots, \gamma_T > 0$ (gradient steps).
1: Set $f_0 \in \arg\min_{g \in \mathcal{F}_0} C(g)$ (initialization).
2: **for** $t = 0$ **to** $T - 1$ **do**
3: Compute $r \leftarrow -\widetilde{\nabla}_n C(f_t)$ (pseudo-residuals).
4: Compute $g_{t+1} \in \arg\min_{g \in \mathcal{F}} \|g(X_1^n) - r\|_2$.
5: Set $f_{t+1} \leftarrow f_t + \gamma_{t+1} g_{t+1}$. (update).
6: **end for**
Output: f_T.

There are several manners to schedule the gradient steps γ_{t+1}, including being adaptively fixed thanks to a line search. This is discussed in Section 5.

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 $\widetilde{\nabla}_n C(f_t)$ instead of the gradient $\nabla_n C(f_t)$. This is, of course, convenient but as explained previously, far from leading to interesting convergence behaviors in practice. 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 $f \in \mathscr{F}_{\mathcal{X}}$, let $\mathrm{Prox}_n^{\lambda} C(f) = \frac{1}{\lambda}\left(f(X_1^n) - \mathrm{prox}_{\lambda D}(f(X_1^n))\right)$, where $\lambda > 0$ is a parameter. The simple idea underlying the proposed algorithm, nicknamed proximal boosting, is that the only difference between subgradient and proximal point methods is the update direction of the optimization variable, which is respectively $\widetilde{\nabla}_n C(f_t)$ or $\mathrm{Prox}_n^{\lambda_{t+1}} C(f_t)$, where $\lambda_{t+1} > 0$ is a proximal step. Thus, proximal boosting computes the pseudo-residuals based on $\mathrm{Prox}_n^{\lambda_{t+1}} C(f_t)$ instead of $\widetilde{\nabla}_n C(f_t)$ and leaves the rest unchanged, as described in Algorithm 2.

While Algorithm 2 is very intuitive and proved to converge at the expected rate for differentiable loss functions (see Section 4), a rate of convergence cannot be exhibited for non-differentiable loss functions. To remedy this limitation, we now introduce a variant of Algorithm 2, named residual proximal boosting (see Algorithm 3) and inspired by [Grubb and Bagnell, 2011], which incorporates a mechanism making it possible to control the approximation error made at each iteration and to obtain a convergence rate under weak assumptions (see Section 4). In practice, it consists in augmenting the pseudo-residuals with the approximation error Δ_t of the previous iteration, also called residual.

As a by-product and along the same line as accelerated gradient boosting [Biau et al., 2019], we remark that this is possible to incorporate Nesterov's acceleration [Nesterov, 1983, Beck and Teboulle,

Algorithm 2 Proximal boosting.

Input: $\lambda_1, \dots, \lambda_T > 0$ (proximal steps).

Set $f_0 \in \arg\min_{g \in \mathcal{F}_0} C(g)$ (initialization).
for $t = 0$ **to** $T - 1$ **do**
  Compute $r \leftarrow -\operatorname{Prox}_n^{\lambda_{t+1}} C(f_t)$ (pseudo-residuals).
  Compute $g_{t+1} \in \arg\min_{g \in \mathcal{F}} \|g(X_1^n) - r\|_2$.
  Set $f_{t+1} \leftarrow f_t + \lambda_{t+1} g_{t+1}$.
end for

Output: f_T.

Algorithm 3 Residual proximal boosting.

Input: $\lambda_1, \dots, \lambda_T > 0$ (proximal steps).

Set $f_0 \in \arg\min_{g \in \mathcal{F}_0} C(g)$, $\Delta_0 \leftarrow 0$ (initialization).
for $t = 0$ **to** $T - 1$ **do**
  Compute $r \leftarrow -\operatorname{Prox}_n^{\lambda_{t+1}} C(f_t)$ (pseudo-residuals).
  Compute $g_{t+1} \in \arg\min_{g \in \mathcal{F}} \|g(X_1^n) - (r + \Delta_t)\|_2$.
  Set $f_{t+1} \leftarrow f_t + \lambda_{t+1} g_{t+1}$.
  Set $\Delta_{t+1} \leftarrow r + \Delta_t - g_{t+1}(X_1^n)$.
end for

Output: f_T.

2009] to proximal boosting, in order to speed up the convergence and to aggregate less weak learners. In practice, Algorithm 4 is similar to Algorithm 2 but computes the proximal step at the auxiliary function h_t instead of f_t. h_{t+1} is then obtained by a momentum tuned by the coefficient α_t, defined recursively by

$$\begin{cases} \beta_0 = 0 \\ \beta_{t+1} = \frac{1+\sqrt{1+4\beta_t^2}}{2}, t \in \mathbb{N} \\ \alpha_{t+1} = \frac{\beta_t - 1}{\beta_{t+1}}, t \in \mathbb{N}. \end{cases} \tag{3}$$

Algorithm 4 returns an estimator $f_T = \sum_{t=0}^{T} w_t g_t$ where the weights $w_0, \dots, w_T$ are now given by a recursive formula (see Appendix C).

The convergence rate of the accelerated proximal point method is $O(1/t^2)$, which prevails over that of the vanilla version of the proximal point method from an optimization point of view. However, as it will be observed in Section 5, the boosting procedure proposed in Algorithm 4 inherits the same drawbacks as accelerated gradient boosting and does not seem reliable. Importantly, it is prone to divergence.

Algorithm 4 Accelerated proximal boosting.

Input: $\lambda_1, \dots, \lambda_T > 0$ (proximal steps).

Set $f_0 = h_0 \in \arg\min_{g \in \mathcal{F}_0} C(g)$ (initialization).
for $t = 0$ **to** $T - 1$ **do**
  Compute $r \leftarrow -\operatorname{Prox}_n^{\lambda_{t+1}} C(h_t)$ (pseudo-residuals).
  Compute $g_{t+1} \in \arg\min_{g \in \mathcal{F}} \|g(X_1^n) - r\|_2$.
  Set $f_{t+1} \leftarrow h_t + \lambda_{t+1} g_{t+1}$.
  Set $h_{t+1} \leftarrow f_{t+1} + \alpha_{t+1}(f_{t+1} - f_t)$.
end for

Output: f_T.

4 Convergence results

This section is dedicated to the theoretical convergence of the two proposed algorithms: proximal boosting (Algorithm 2) and residual proximal boosting (Algorithm 3).

A preliminary result on the convergence of the proximal boosting technique can be easily derived upon previous work by Rockafellar [1976]: it requires to control the error introduced by considering an approximated direction of optimization instead of the true proximal step, and could be stated as follows in the case of Algorithm 2.

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

$$\varepsilon_{t+1} = \left\| g_{t+1}(X_1^n) + \mathrm{Prox}_n^{\lambda_{t+1}} C(f_t) \right\|_2 .$$

Suppose that $(f_t(X_1^n))_t$ is bounded and that

$$\sum_{t=0}^{+\infty} \varepsilon_t < +\infty. \tag{4}$$

Then,

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

Theorem 1 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 for Algorithm 2. This is the role of the following assumption, which is common in the boosting literature to characterize the approximation capacity of the class $\mathcal{F}$ [Grubb and Bagnell, 2011].

(A) There exists $\zeta \in (0, 1]$ such that:

$$\forall r \in \mathbb{R}^n, \qquad \exists g \in \mathcal{F} : \|g(X_1^n) - r\|_2^2 \leq (1 - \zeta^2) \|r\|_2^2 .$$

A set of weak learners $\mathcal{F}$ satisfying Assumption (A) is said to have edge ζ.

Now, we provide a convergence result for Algorithm 2, based on smoothness properties: a functional C of the form $C(f) = D(f(X_1^n))$, for all $f \in \mathscr{F}_{\mathcal{X}}$, is said L-smooth (for some $L > 0$) if D is differentiable and for all $x, x' \in \mathbb{R}^n$,

$$D(x') \leq D(x) + \langle \nabla D(x), x' - x \rangle + \frac{L}{2} \|x' - x\|_2^2 ,$$

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

$$D(x') \geq D(x) + \langle \nabla D(x), x' - x \rangle + \frac{\kappa}{2} \|x' - x\|_2^2 ,$$

where $\langle \cdot, \cdot \rangle$ refers to the inner product. The convergence rate stated hereafter is based on an original result presented and proved in Appendix A.

Theorem 2. *Assume that (A) is granted, 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 $f^\star \in \arg\min_{f \in \operatorname{span} \mathcal{F}} C(f)$. Then, choosing $\lambda_t = \frac{\zeta^2}{8L}$ leads to:*

$$C(f_T) - C(f^\star) \leq \left(1 - \frac{\zeta^4 \kappa}{21 L}\right)^T \left(C(f_0) - C(f^\star)\right).$$

Proof. Given that $\forall f \in \mathscr{F}_{\mathcal{X}} : C(f) = D(f(X_1^n))$ and Assumptions (E), (SM) and (SC) are granted for D (respectively by Assumption (A), L-smoothness and κ-strong convexity of C), this is an application of Theorem 4 (see Appendix A) to the function D. □

Theorem 2 states that proximal boosting has a linear convergence rate under smoothness and strong convexity assumptions. This result was expected since gradient boosting has the same convergence rate under these assumptions [Grubb and Bagnell, 2011].

Admittedly, these two assumptions are restrictive for an algorithm designed for non-differentiable loss functions. However, our analysis revealed that they seem necessary to control the impact of the approximation error on the convergence. Consequently, proving convergence for proximal boosting under weaker assumptions on the objective function C (see thereafter) requires to modify Algorithm 2. This is the role of Algorithm 3, as introduced in Section 3.

A functional C of the form $C(f) = D(f(X_1^n))$, for all $f \in \mathscr{F}_{\mathcal{X}}$, is said to be G-Lipschitz continuous (for some $G > 0$) if for all $x, x' \in \mathbb{R}^n$,

$$|D(x) - D(x')| \leq G \left\| x - x' \right\|_2 .$$

A convergence rate for residual proximal boosting (Algorithm 3) can be derived from this weak property, as stated in Theorem 3 (which is based on an original result presented and proved in Appendix A).

Theorem 3. *Assume that (A) is granted, C is convex and G-Lipschitz continuous for some $G > 0$. Let $(f_t)_t$ be any sequence generated by Algorithm 3 and $f_{best} \in \arg\min_{1 \leq t \leq T} C(f_t)$. Assume that there exists $f^\star \in \arg\min_{f \in \operatorname{span} \mathcal{F}} C(f)$ and that $\|f_t(X_1^n)\|_2 \leq R$ and $\|f^\star(X_1^n)\|_2 \leq R$ for some $R > 0$ and all t. Then, choosing $\lambda_t = \frac{1}{\sqrt{t}}$ leads to:*

$$C(f_{best}) - C(f^\star) \leq \frac{2R^2}{\sqrt{T}} + \frac{40G^2}{\zeta^4 \sqrt{T}} + \frac{2G^2}{\zeta^4 T^{\frac{3}{2}}}.$$

Proof. Given that $\forall f \in \mathscr{F}_{\mathcal{X}} : C(f) = D(f(X_1^n))$ and Assumptions (E) and (L) are granted for D (respectively by Assumption (A) and G-Lipschitz continuity of C), this is an application of Theorem 6 (see Appendix A) to the function D. □

Theorem 3 states that the best aggregation returned by residual proximal boosting has sublinear convergence rate (more precisely $O(1/\sqrt{t})$) under Lipschitz continuity assumption. On the one hand, this rate is similar to that of residual gradient boosting [Grubb and Bagnell, 2011], showing that our approach is theoretically competitive with the state-of-the art regarding boosting with non-differentiable cost functions. On the other hand, this result is quite pessimistic regarding the empirical performance of Algorithm 3: Section 5 will show that, in practice, linear convergence (as stated by Theorem 2) is often observed numerically, even though the loss function is not differentiable. This is perfectly consistent with our initial intuition: boosting better handles the non-differentiability of the objective function by using the proximal operator instead of any subgradient.

Remark 4.1. *Since the convergence rate of the proximal point method for non-smooth functions is $O(1/t)$ (respectively $O(1/\sqrt{t})$ for the subgradient method), one may expect that proximal boosting converges in $O(1/t)$ (while subgradient boosting is in $O(1/\sqrt{t})$ [Grubb and Bagnell, 2011]) but the previous result states a worst case convergence rate in $O(1/\sqrt{t})$.*

The latter is in fact not that surprising: for L-smooth and κ-strongly convex objectives, gradient descent converges in $O\left(\left(1 - \frac{\kappa}{L}\right)^t\right)$ while gradient boosting converges in $O(1/t)$. This highlights that the approximation step (represented by the operator P below) used in boosting iterations is prone to damage the convergence rate.

More formally, consider an objective function f and two iterations $x_{t+1}^{theo} = x_t - \gamma_t d_t$ and $x_{t+1} = x_t - \gamma_t P(d_t)$, where P is an approximation operator. The rate in $O(1/t)$ for gradient descent and the proximal point method is linked to the capability to control the error $\varepsilon(\tilde{\nabla} f(x_{t+1}^{theo}), d_t)$ between a subgradient at x_{t+1}^{theo}, denoted $\tilde{\nabla} f(x_{t+1}^{theo})$, and the direction of descent at x_t, denoted d_t.

The error $\varepsilon(\tilde{\nabla} f(x_{t+1}^{theo}), d_t)$ is (i) controlled under the assumption of Lipschitz continuous gradients in the case of gradient descent; (ii) equal to 0, ($\varepsilon(\tilde{\nabla} f(x_{t+1}^{theo}), d_t) = 0$) for the proximal point method (the proximal direction of descent is exactly a subgradient at x_{t+1}^{theo}). If this error cannot be controlled tightly, we may end up with a $O(1/\sqrt{t})$ convergence rate. This is exactly the case for the subgradient method, for which $\varepsilon(\tilde{\nabla} f(x_{t+1}^{theo}), d_t) = \varepsilon(\tilde{\nabla} f(x_{t+1}^{theo}), \tilde{\nabla} f(x_t))$ (which is a difference between two subgradients) is only bounded by a constant.

In proximal and subgradient boosting (second iteration as defined above), this error can be decomposed into three parts:

$$\varepsilon(\tilde{\nabla} f(x_{t+1}), P(d_t)) = \varepsilon(\tilde{\nabla} f(x_{t+1}), \tilde{\nabla} f(x_{t+1}^{theo})) + \varepsilon(\tilde{\nabla} f(x_{t+1}^{theo}), d_t) + \varepsilon(d_t, P(d_t)).$$

1. *Without strong assumptions on f, the first term (being a difference between two subgradients) is of the order of $O(1/\sqrt{t})$.*
2. *The second term can be controlled by $O(1/\sqrt{t})$ when $d_t = \tilde{\nabla} f(x_t)$ (subgradient boosting) and 0 if d_t is the proximal direction (proximal boosting).*
3. *The third term usually requires the edge hypothesis (as used in the literature and in this paper) and a specific mechanism inside the algorithm to be controlled, such as the residual inspired by [Grubb and Bagnell, 2011].*

Overall, even if proximal boosting benefits from the cancellation of the second error term, the first and the third ones remain limiting, resulting in an $O(1/\sqrt{t})$ rate, as in the case of subgradient boosting.

5 Numerical analysis

In Section 3, proximal boosting algorithms have been introduced in a fairly general way. However, the empirical results presented in this section are based on an implementation (See Algorithm 5) incorporating some modifications that have made the success of gradient boosting.

First of all, the proximal step is fixed to some positive value: $\lambda_t = \lambda > 0$; and the update rule $f_{t+1} \leftarrow f_t + \lambda_{t+1} g_{t+1}$ is replaced by $f_{t+1} \leftarrow f_t + \nu \gamma_{t+1} g_{t+1}$, where

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

In other words, the step size is tuned by a shrinkage coefficient (or learning rate) $\nu \in (0, 1]$ and a line search producing the largest decrease of the objective function.

The learning rate is known to be a key element of boosting machines in order to obtain a good generalization performance. To understand that fact, let us remark that the number of iterations T 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 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.

Besides the learning rate, the step size is controlled by a line search, that simply scales the weak learner g_{t+1} by a constant factor. Actually, since the class of weak learners $\mathcal{F}$ is in practice a set of regression trees (implemented in Scikit-learn [Pedregosa et al., 2011]), a multiple line search is used, as proposed by Friedman [2001]: a line search is performed sequentially for each leaf of the decision tree, such that each level of the piecewise constant function g_{t+1} is scaled with its own factor. All variants of proximal and gradient boosting are implemented based on Algorithm 5 in the Scikit-learn fashion [Pedregosa et al., 2011] and are freely available in the Python package `optboosting`[1].

[1] `https://github.com/msangnier/optboosting`

Algorithm 5 Meta-algorithm for boosting.

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

1: Set $f_0 \in \arg\min_{g \in \mathcal{F}_0} C(g)$, $\Delta_0 \leftarrow 0$ (initialization).

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

3: Pseudo-residuals (see Appendix B):

$$\begin{cases} r \leftarrow -\widetilde{\nabla}_n C(f_t) & \text{for gradient boosting,} \\ r \leftarrow -\operatorname{Prox}_n^{\lambda} C(f_t) & \text{for proximal boosting.} \end{cases}$$

4: Regression of the vector $r + \Delta_t \in \mathbb{R}^n$ onto $(X_1, \ldots, X_n)$:

$$g_{t+1} \in \arg\min_{g \in \mathcal{F}} \|g(X_1^n) - (r + \Delta_t)\|_2 .$$

5: Residual:

$$\begin{cases} \Delta_{t+1} \leftarrow 0 & \text{for vanilla boosting,} \\ \Delta_{t+1} \leftarrow r + \Delta_t - g_{t+1}(X_1^n) & \text{for residual boosting.} \end{cases}$$

6: Line-search (see Appendix B):

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

7: Update:

$$f_{t+1} \leftarrow f_t + \nu \gamma_{t+1} g_{t+1}.$$

8: **end for**

Output: f_T.

5.1 Behavior of proximal boosting

Based on synthetic data, this section aims at numerically illustrating the performance of proximal boosting compared to gradient boosting. For this purpose, two synthetic models are studied, both coming from Biau et al. [2019, 2016]:

Regression:

$$\left\| \begin{array}{l} n = 800, d = 100; \\ Y = -\sin(2X^{(1)}) + {X^{(2)}}^2 + X^{(3)} - \exp(-X^{(4)}) + Z_{0.5}, \end{array} \right.$$

Classification:

$$\left\| \begin{array}{l} n = 1500, d = 50; \\ Y = \begin{cases} 1 & \text{if } X^{(1)} + {X^{(4)}}^3 + X^{(9)} + \sin(X^{(12)} X^{(18)}) + Z_{0.1} > 0.38; \\ -1 & \text{otherwise,} \end{cases} \end{array} \right.$$

where Z_{σ^2} is a random variable independent from X, following a normal distribution with zero mean and variance σ^2.

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 $(X^{(j)})_{1 \le j \le d}$, are normally distributed with zero mean and covariance matrix $\Sigma = \left(2^{-|i-j|}\right)_{1 \le i,j \le d}$. Moreover, in these synthetic models of regression and classification, an additive and independent noise is embodied by the random variable Z_{σ^2}.

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	$\beta > 0$	$\exp(-\beta y y')$	Classification
Logistic	-	$\log_2(1+\exp(-yy'))$	Classification
Hinge	-	$\max(0, 1-yy')$	Classification

Table 2: Loss functions.

Four different losses are considered (see Table 2 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 B. On that occasion, it can be remarked that the direction of descent $\text{Prox}_n^\lambda C(f_t)$ of proximal boosting applied with the least squares loss is the same as that of gradient boosting, $\nabla_n C(f_t)$, up to a constant factor (see Appendix B). 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 2). Nevertheless, since the numerical behaviors are respectively very close to the least absolute deviations and the exponential cases, the results are not reported.

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 test set (50%). The performance of the methods are appraised through several curves representing the training and test losses along the $T = 1000$ iterations of boosting.

5.1.1 Convergence

As a first numerical experiment, we aim at illustrating the convergence of proximal boosting (see Section 4) for two classes $\mathcal{F}$ of weak learners: regression trees with maximal depth 3 (in blue in Figure 2) and with maximal depth 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.

For the purpose of the analysis, parameters λ and ν are set to standard values: $\lambda = 1$, $\nu = 5 \cdot 10^{-2}$, which does not hurt the generality of the forthcoming interpretations. Moreover, gradient boosting and its variant proposed by Grubb and Bagnell [2011], residual gradient boosting, are included as references.

Let us analyze the top panels of Figure 2: for differentiable losses (least squares and exponential), proximal and gradient descents behave exactly the same (curves with symbols **P** and **G** are mixed up). Moreover, as theoretically analyzed in Theorem 2, the rate of convergence of proximal boosting is linear with a slope that increases with the capacity of the class of weak learners (even though the exponential loss is not strongly convex).

Still for differentiable losses, the use of the residual originally introduced to derive a convergence rate under weak assumptions (represented with dotted lines and symbols **RP** and **RG** 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).

Concerning non-differentiable losses (least absolute deviations and hinge on the bottom panels of Figure 2), proximal boosting converges faster than gradient boosting, which does not seem to converge for the hinge loss. In addition, it is noticeable to observe that convergence of proximal boosting seems almost linear while the empirical risk violates the assumptions of smoothness required for Theorem 2.

For non-differentiable losses, the use of the residual helps gradient boosting to converge. Yet, we remark that residual proximal boosting behaves similarly to proximal boosting (curves with symbols

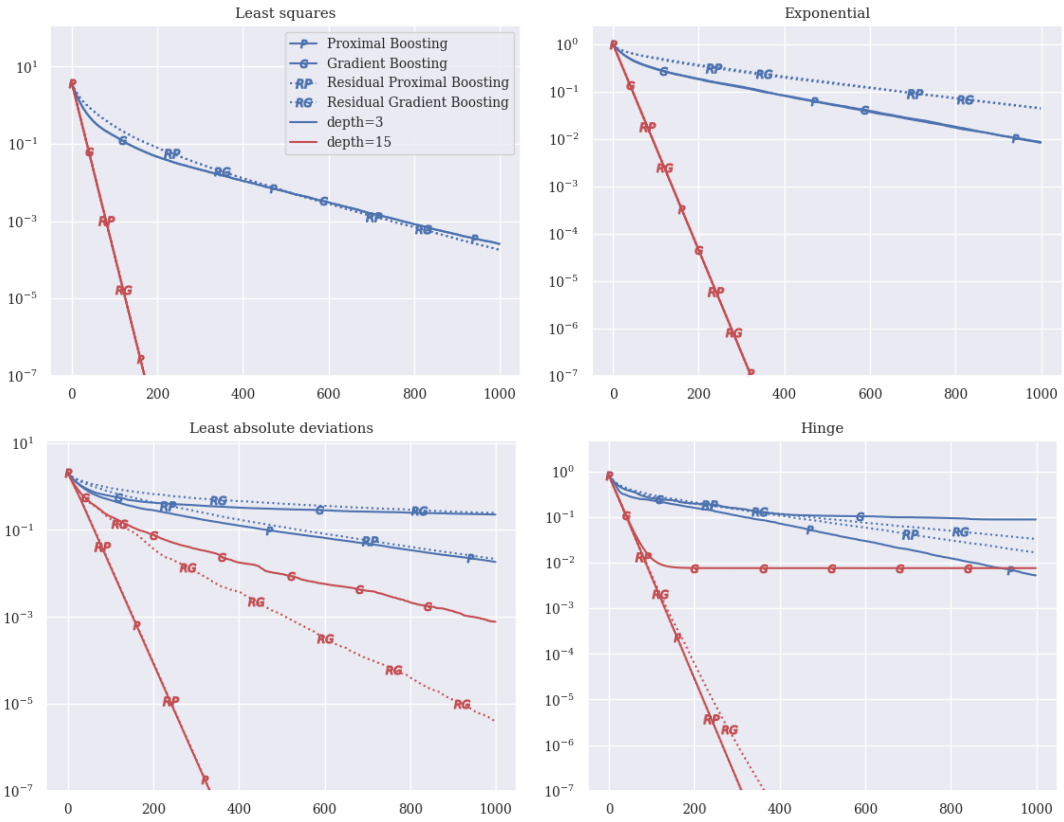

Figure 2: Training losses for two values of maximal depth (3 in blue, 15 in red) vs number of iterations on the horizontal axis.

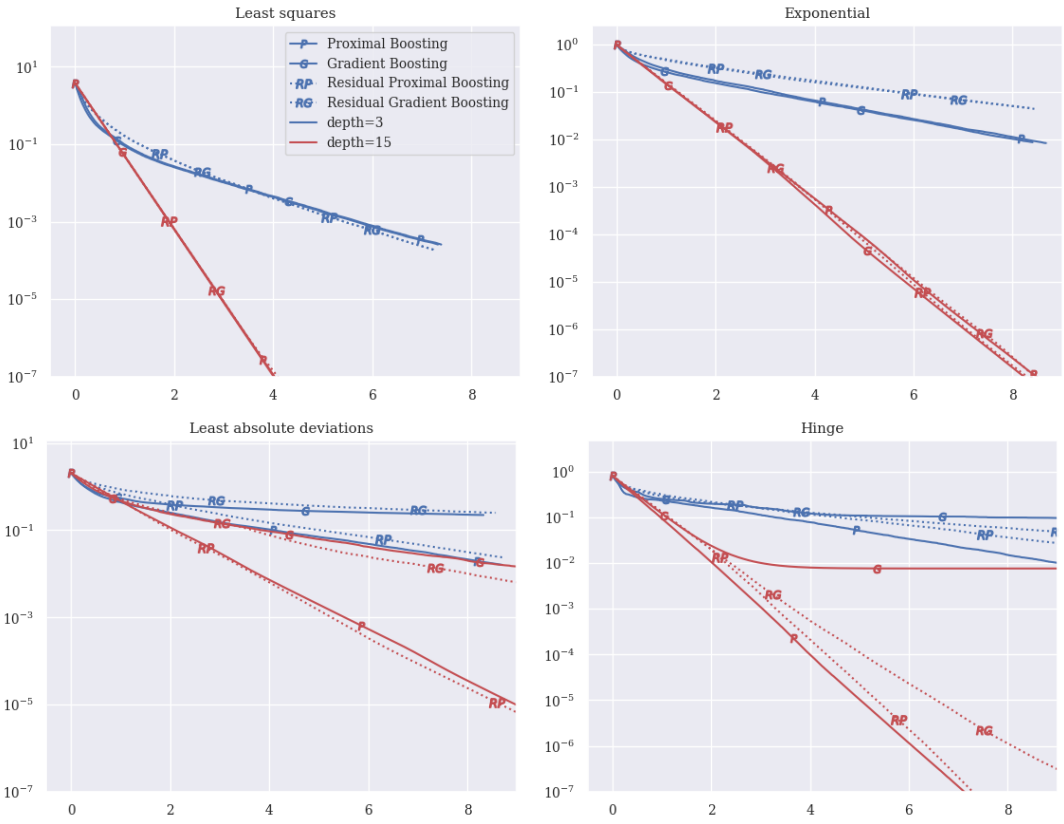

Figure 3: Training losses for two values of maximal depth (3 in blue, 15 in red) vs clock time (seconds) on the horizontal axis.

RP and **P** are mixed up), suggesting that, from a convergence point of view, this mechanism is more needed for a theoretical purpose than for a practical one.

As a last piece of evidence, Figure 3 depicts the same experiment as the previous one but with the clock time on the horizontal axis. We can remark that the behavior of algorithms is similar when convergence is analyzed with respect to the number of iterations or to the time elapsed. This shows that computing a proximal direction of descent is not more expensive than computing a (sub)gradient, which is in favor of proximal boosting.

Overall, this numerical experiment confirms the initial intuition that proximal boosting behaves better than gradient boosting and residual gradient boosting in the non-differentiable cases. Keeping in mind that behaviors are similar for differentiable losses, we carry on the study only with least absolute deviations and hinge losses.

5.1.2 Proximal step

We aim at illustrating the impact of the proximal step λ intervening in proximal boosting as a new parameter. For this purpose, Figure 4 depicts the trend of training (top) and test (bottom) losses of proximal boosting for $\lambda \in \{10^{-2}, 10^{-1}, \dots, 10^{2}\}$ (see the different colors) and decision trees of maximal depth 3 as weak learners. Compared algorithms include proximal (Algorithm 2), residual proximal (Algorithm 3) and accelerated proximal (Algorithm 4) boosting, as well as their gradient counterparts (in black, independent of λ).

Figure 4 shines a light of the tie between the proximal step and the convergence rate: the bigger λ, the faster the convergence of the training and test losses. As a consequence (see the top panel), proximal

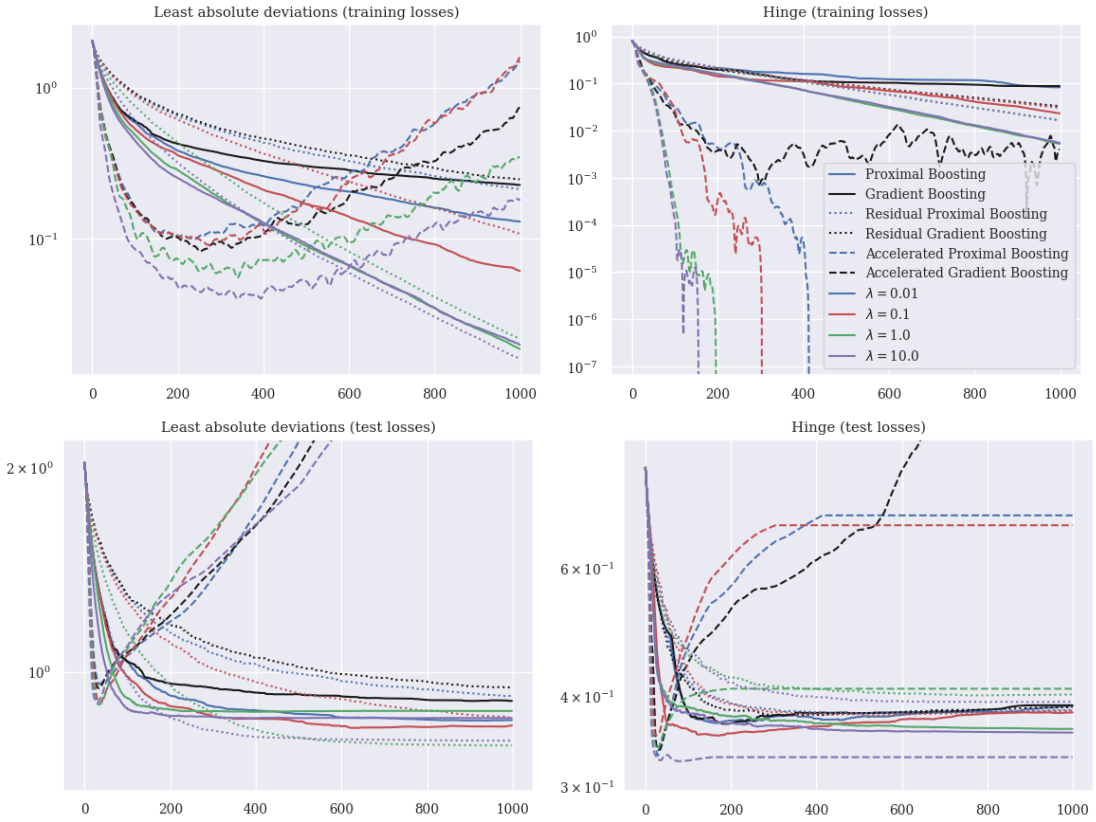

Figure 4: Training (top) and test (bottom) losses of proximal boosting algorithms for several values of the proximal step λ vs number of iterations on the horizontal axis.

boosting prevails over gradient boosting from an optimization perspective because it converges faster for sufficiently large λ. Regarding the training loss, the advantage of using the residual is not clear since proximal boosting offers similar convergence rates than residual proximal boosting for large values of λ, and converges faster than residual gradient boosting.

Analyzing the test loss, proximal and residual proximal boosting achieve lower errors than gradient and residual gradient boosting for intermediate values of λ (between 0.1 and 10). In addition, their behavior is quite stable with respect to the parameters from our experience.

From all points of view, using a proximal direction of descent is a real advantage over subgradient. Besides, from a global perspective, proximal boosting helps to build more accurate models than gradient boosting.

This numerical experiment is also a place for studying the benefit of incorporating Nesterov's acceleration into boosting. For proximal as well as gradient boosting, acceleration speeds up the decrease of the training and test losses, and thus it makes it possible to build boosted models with very few weak learners. Nevertheless, both accelerated boosting approaches suffer from instabilities leading to divergence, on the training and on the test sets. Regarding the test error, they do not seem to be capable to produce very accurate models (on the bottom panel of Figure 4, non-accelerated methods offer a lower test error than accelerated ones). Remark that, accelerated proximal boosting performs definitely better than its accelerated gradient counterpart. We guess that

1. the procedure is diverging because Nesterov's extrapolation intensifies the boosting approximation error made at each iteration;

2. the acceleration makes the method very sensitive and dependent on a fine tuning procedure in order to perform well in generalization.

Even though divergence on the training error seems to always occur in the overfitting regime (i.e. after the minimal test error, see Figure 4), these observations are not in line with a statistically reliable learning technique. As a consequence, such methods are only recommended to build models with very few trees, for instance because of hardware constraints.

5.2 Generalization in real world cases

This section aims at comparing the generalization ability of the proposed boosting estimators with respect to variants of gradient boosting, as well as extreme gradient boosting (XGBoost) [Chen and Guestrin, 2016] and random forests [Breiman, 2001]. The last two methods are introduced in the numerical comparison only as benchmarks. Indeed, random forests aggregate weak learners but with equal weights, and XGBoost is a boosting method based on second order optimization. From a strict optimization point of view, second order optimization is not applicable to non-differentiable loss functions, nevertheless, given the liberty taken with Nesterov's acceleration, XGBoost is applied as a black box for minimizing the empirical loss. It is important to point out that, up to our knowledge, there is no convergence result for XGBoost with non-differentiable losses.

Comparison is based on nine datasets (available on the UCI Machine Learning repository), the characteristics of which are described in Table 3. The first six 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 $T \in [1, 1000]$, maximal depth of decision trees varying in $[1, 3, 5]$, learning rate $\nu \in \{5 \cdot 10^{-2}, 10^{-1}, 3 \cdot 10^{-1}, 5 \cdot 10^{-1}, 1\}$ and proximal step $\lambda \in \{10^{-3}, 10^{-2}, \ldots, 10^{2}\}$ for boosting, completed with the maximal number of features for random forests) 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 is 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
Whitewine	4898	11	Regression
Redwine	1599	11	Regression
BostonHousing	506	13	Regression
Crabs	200	4	Regression
Engel	235	1	Regression
Sniffer	125	4	Regression
Adult	30162	13	Classification
Advertisements	2359	1558	Classification
Spam	4601	57	Classification

Table 3: Real-world datasets (n: sample size, d: number of attributes).

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 2 for a quick definition and Appendix B for the details). Since random forests are not explicitly designed for minimizing theses losses, only the least squares test loss and the classification error are reported.

5.2.1 Regression problems

Test losses for the least squares (top), least absolute deviations (middle) and pinball (bottom) losses are described in Figure 5. Δ Test loss refers to the increment of the loss from that of gradient boosting.

Regarding the least squares setting, let us remind that gradient and proximal boosting boil down to be the same method (the directions of descent are exactly the same). We observe that they achieve a performance comparable to extreme gradient boosting and better than random forests. Moreover, even though residual boosting was not designed for differentiable losses, it provides the most accurate models for 3 datasets out of 6.

Looking now at least absolute deviations and pinball losses, we observe that proximal boosting always achieves better predictions than gradient boosting. In addition, in the bulk of the situations, the most accurate method is either proximal or residual proximal boosting. This confirms our intuition concerning the need for optimization techniques suited for non-differentiable loss functions.

Regarding accelerated versions of boosting, as expected they do not produce more accurate models than vanilla boosting, very likely because convergence is so fast that tuning parameters becomes excessively tricky. Incidentally, we remark that accelerated proximal boosting offers better generalization performances than accelerated gradient boosting for non-differentiable losses (except for the dataset Engel).

5.2.2 Classification problems

Losses and misclassification rates computed on the test datasets are depicted respectively in Figure 6 and in Figure 7 for the exponential (top) and the hinge (bottom) losses. Besides Δ Test loss/error, referring to the increment of the loss or misclassification rate from that of gradient boosting, Hinge-Exponential in Figure 7 represents the increment of the misclassification rate of hinge loss-based boosting from that obtained with the exponential loss.

Regarding both indicators (loss in Figure 6 and error in Figure 7), four methods share the winners' podium: proximal boosting (blue), residual proximal boosting (orange), residual gradient boosting (red) and XGBoost (pink). For the hinge loss, proximal or residual proximal boosting are always better than gradient and residual gradient boosting. Moreover, accelerated proximal boosting (purple) always gives better loss and accuracy than gradient and accelerated gradient boosting (brown). Both observations confirm the interest of proximal-based boosting for non-differentiable losses.

It is remarkable that XGBoost performs quite well with the hinge loss, while it was not originally designed for non-differentiable losses. Nevertheless, the bottom panel of Figure 7 shows that, overall

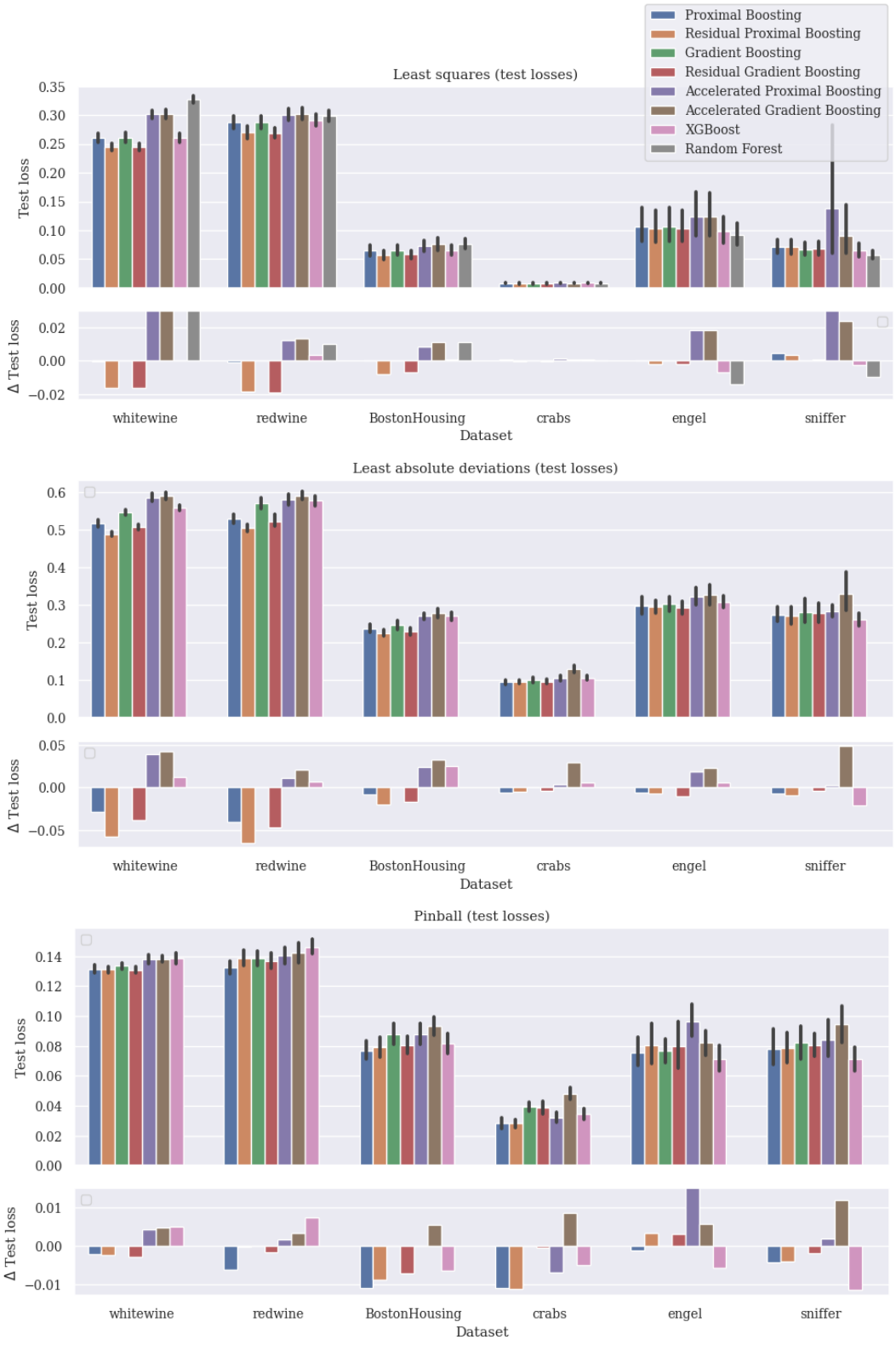

Figure 5: Losses on test datasets for the least squares (top), least absolute deviations (middle) and pinball (bottom) losses. Δ Test loss refers to the increment of the loss from that of gradient boosting. The methods proposed in this article are in blue, orange and green.

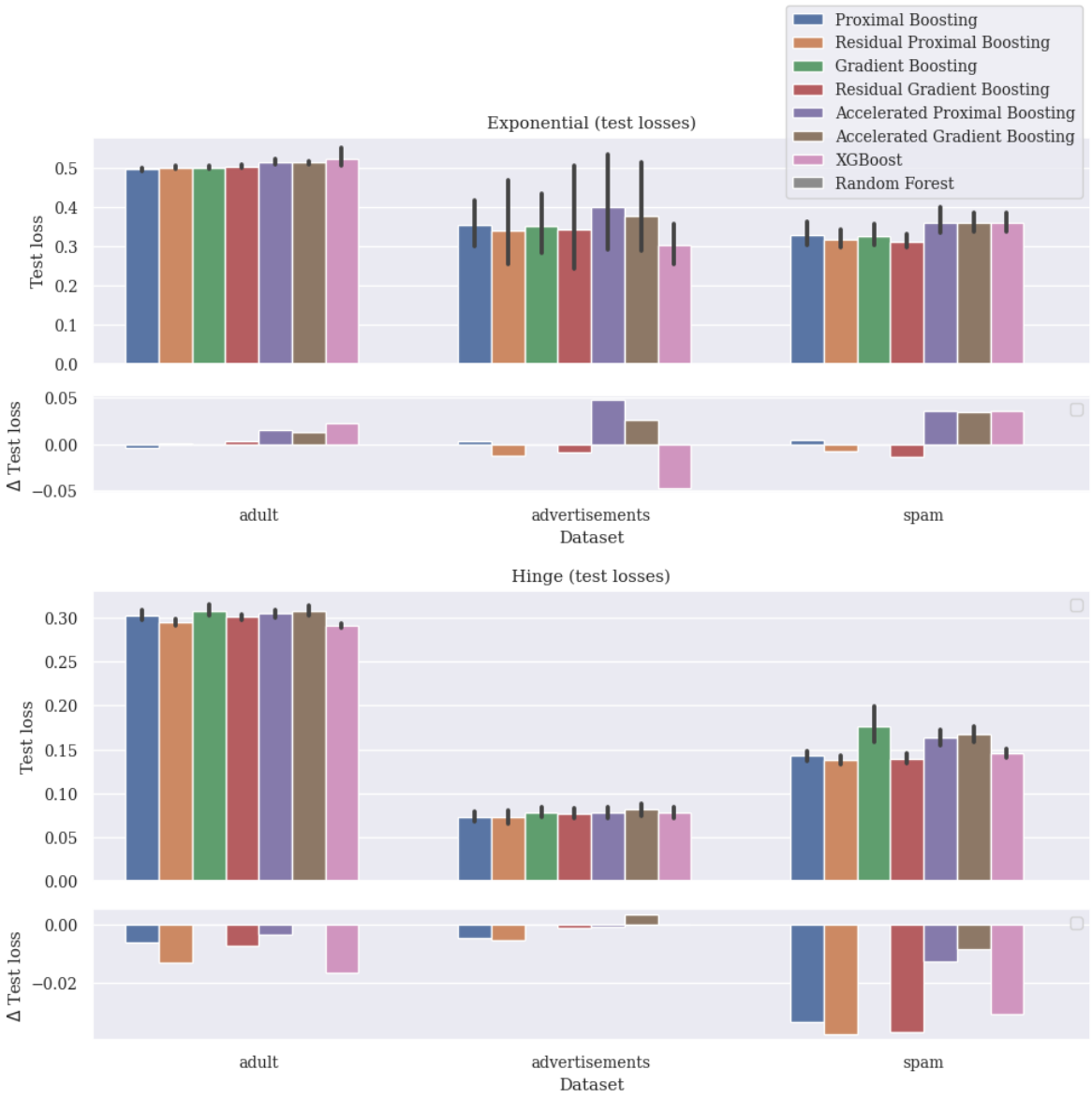

Figure 6: Losses on test datasets for the exponential (top) and hinge (bottom) losses. Δ Test loss refers to the increment of the loss from that of gradient boosting. The methods proposed in this article are in blue, orange and green.

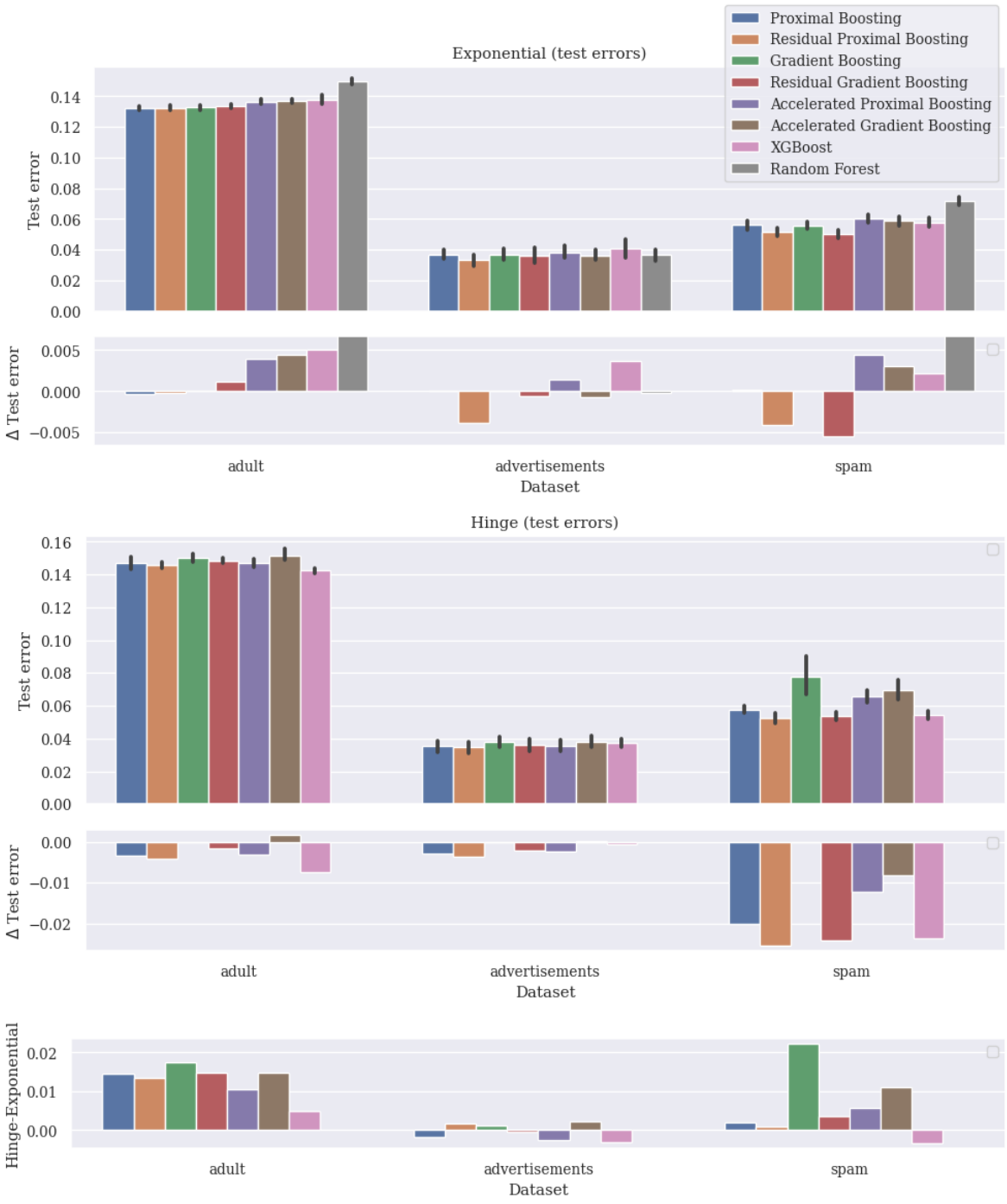

Figure 7: Misclassification rates on test datasets for the exponential (top) and hinge (bottom) losses. Δ Test error and Hinge-Exponential refer to the increment of the misclassification rate respectively from that of gradient boosting and from that obtained with the exponential loss. The methods proposed in this article are in blue, orange and green.

using a hinge loss instead of an exponential loss is rarely a big advantage, except to obtain sporadically a marginal gain in accuracy.

6 Conclusion

Building upon the proximal point method for convex and non-smooth optimization, this paper has introduced two novel boosting algorithms, nicknamed proximal boosting and residual proximal boosting, which have appeal for non-differentiable loss functions ℓ. A theoretical study demonstrates convergence of proximal and residual proximal boosting from an optimization point of view (under different hypotheses on the loss function). Numerical experiments on synthetic data confirm the theoretical convergence results and show a significant impact of the newly introduced parameter λ. Correctly tuned, this parameter provides a noticeable improvement of proximal-based boosting over gradient-based boosting for non-differentiable loss function, from both the optimization and the statistical points of view. Moreover, in real-world regression and classification situations, proximal or residual proximal boosting often achieve the best test loss and are, overall, very competitive with state-of-the-art boosting approaches.

As a by-product, we have also studied incorporating Nesterov's acceleration to proximal boosting, as done with gradient boosting in [Biau et al., 2019]. Numerically, we observe instabilities in both algorithms, leading to divergence on the training and the test sets. Our experience is that accelerated boosting is very sensitive to hyperparameters and thus tricky to tune. Despite the fact that these procedures rarely provide good generalization results, accelerated proximal boosting seems to perform better than its gradient counterpart for non-differentiable losses.

Going further in the theoretical analysis of accelerated proximal boosting is also an exciting perspective. In particular, Lu et al. [2020] recently proposed a variant of accelerated gradient boosting [Biau et al., 2019] with guaranteed convergence for differentiable and smooth losses. Establishing similar results for accelerated proximal boosting constitutes an important challenge both from a numerical and a theoretical point of view.

On another note, 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, just like the Frank-Wolfe algorithm has impacted boosting [Wang et al., 2015, Jaggi, 2013]. This may also hold true for non-differentiable and non-convex optimization (see for instance [Ochs et al., 2014]).

Acknowledgements

The authors are thankful to Gérard Biau and Jalal Fadili for enlightening discussions. They also thank the three anonymous referees for their constructive comments.

References

Md. M. Ahamad, S. Aktar, Md. Rashed-Al-Mahfuz, S. Uddin, P. Liò, H. Xu, M. A. Summers, Julian M. W. Quinn, and M. A. Moni. A machine learning model to identify early stage symptoms of SARS-Cov-2 infected patients. Expert Systems with Applications, 160:113661, 2020.

Md. A. Awal, M. Masud, Md. S. Hossain, A. A.-M. Bulbul, S. M. H. Mahmud, and A. K. Bairagi. A Novel Bayesian Optimization-Based Machine Learning Framework for COVID-19 Detection From Inpatient Facility Data. IEEE Access, 9:10263–10281, 2021.

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, pages 23–44. Springer International Publishing, Cham, 2021.

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. Machine Learning, 108(6): 971–992, 2019. ISSN 0885-6125.

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.

P. Bühlmann and B. Yu. Boosting With the L2 Loss. Journal of the American Statistical Association, 98(462):324–339, 2003.

Y. Cai, H. Hang, H. Yang, and Z. Lin. Boosted Histogram Transform for Regression. In Proceedings of the 37th International Conference on Machine Learning, pages 1251–1261. PMLR, 2020.

J. Chen, Z. Chu, R. Zhao, A. F. Luo, and K. H. Luo. Output prediction of alpha-type Stirling engines using gradient boosted regression trees and corresponding heat recovery system optimization based on improved NSGA-II. Energy Reports, 8:835–846, 2022.

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.

J. Cui, H. Hang, Y. Wang, and Z. Lin. GBHT: Gradient Boosting Histogram Transform for Density Estimation. In Proceedings of the 38th International Conference on Machine Learning, pages 2233–2243. PMLR, 2021.

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.

T. Gao and D. Koller. Multiclass boosting with hinge loss based on output coding. In Proceedings of the 28th International Conference on International Conference on Machine Learning, pages 569–576, Madison, WI, USA, 2011. Omnipress.

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.

H. Hang, T. Huang, Y. Cai, H. Yang, and Z. Lin. Gradient Boosted Binary Histogram Ensemble for Large-scale Regression. arXiv:2106.01986 [cs, stat], 2021.

C. C. Ikeagwuani, D. C. Nwonu, and C. C. Nweke. Resilient modulus descriptive analysis and estimation for fine-grained soils using multivariate and machine learning methods. International Journal of Pavement Engineering, pages 1–16, 2021.

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.

S.-B. Lin, Y. Lei, and D.-X. Zhou. Boosted Kernel Ridge Regression: Optimal Learning Rates and Early Stopping. Journal of Machine Learning Research, 20(46):1–36, 2019.

H. Lu, S. P. Karimireddy, N. Ponomareva, and V. Mirrokni. Accelerating Gradient Boosting Machines. In Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, pages 516–526, Online, 2020. PMLR.

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 $O(1/k^2)$. 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.

B. U. Park, Y. K. Lee, and S. Ha. L_2 boosting in kernel regression. Bernoulli, 15(3):599–613, 2009.

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.

S. Rajendran, S. Srinivas, and T. Grimshaw. Predicting demand for air taxi urban aviation services using machine learning algorithms. Journal of Air Transport Management, 92:102043, 2021.

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.

I. V. D. S. Santana, A. C. M. Silveira, A. Sobrinho, L. C. Silva, L. D. Silva, D. F. S. Santos, E. C. Gurjão, and A. Perkusich. Classification Models for COVID-19 Test Prioritization in Brazil: Machine Learning Approach. Journal of Medical Internet Research, 23(4):e27293, 2021.

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

V. N. Temlyakov. Greedy expansions in convex optimization. Proceedings of the Steklov Institute of Mathematics, 284:244–262, 2014.

H. Tyralis and G. Papacharalampous. Boosting algorithms in energy research: a systematic review. Neural Computing and Applications, 33(21):14101–14117, 2021.

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

Y. Wang, X. Liao, and S. Lin. Rescaled Boosting in Classification. IEEE Transactions on Neural Networks and Learning Systems, 30(9):2598–2610, 2019.

J. Zeng, M. Zhang, and S.-B. Lin. Fully corrective gradient boosting with squared hinge: Fast learning rates and early stopping. Neural Networks, 147:136–151, 2022.

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, 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.

T. Zhang and B. Yu. Boosting with early stopping: Convergence and consistency. The Annals of Statistics, 33(4):1538–1579, 2005.

A Analysis of the approximated proximal point method

A.1 Setting

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 convex.

For an operator $P : \mathbb{R}^n \to \mathbb{R}^n$, we consider the approximated proximal point method, described in Algorithm 6, as well as the approximated proximal point method with accumulation, described in Algorithm 7. Both are similar to the proximal point iteration but makes use of a modified direction of update ($P(g_t)$ or $P(g_t + \Delta_t)$ instead of g_t). In particular, let us remark that when $P(x) = x$, Algorithms 6 and 7 recover the original proximal point method.

Algorithm 6 Approximated proximal point method.

Input: T (number of iterations), $\lambda_0, \dots, \lambda_{T-1} > 0$ (proximal steps), $P : \mathbb{R}^n \to \mathbb{R}^n$ (approximation operator).

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

Output: x_T.

Algorithm 7 Approximated proximal point method with accumulation.

Input: T (number of iterations), $\lambda_0, \dots, \lambda_{T-1} > 0$ (proximal steps), $P : \mathbb{R}^n \to \mathbb{R}^n$ (approximation operator).

1: Set $x_0 \in \mathbb{R}^n$ and $\Delta_0 = 0$ (initialization).
2: **for** $t = 0$ **to** $T - 1$ **do**
3: $\quad g_t \leftarrow \frac{1}{\lambda_t}\left(x_t - \operatorname{prox}_{\lambda_t F}(x_t)\right)$.
4: $\quad x_{t+1} \leftarrow x_t - \lambda_t P(g_t + \Delta_t)$.
5: $\quad \Delta_{t+1} = g_t + \Delta_t - P(g_t + \Delta_t)$.
6: **end for**

Output: x_T.

The forthcoming sections prove convergence of Algorithm 6 for strongly convex functions with Lipschitz continuous gradient (linear rate exhibited in Theorem 4) and of Algorithm 7 for Lipschitz continuous functions (sublinear rate exhibited in Theorem 6). To be more formal, the following assumptions will be used:

(SM) F is L-smooth (for some $L > 0$): F is differentiable and

$$\forall x, x' \in \mathbb{R}^n, \qquad F(x') \leq F(x) + \langle \nabla F(x), x' - x \rangle + \frac{L}{2} \|x' - x\|_2^2 .$$

(SC) F is κ-strongly convex (for some $\kappa > 0$):

$$\forall x, x' \in \mathbb{R}^n, \forall \eta \in \partial F(x), \qquad F(x') \geq F(x) + \langle \eta, x' - x \rangle + \frac{\kappa}{2} \|x' - x\|_2^2 .$$

(L) F is G-Lipschitz continuous (for some $G > 0$):

$$\forall x \in \mathbb{R}^n, \forall \eta \in \partial F(x), \qquad \|\eta\|_2 \leq G.$$

In any case, it is assumed that:

(E) There exists $\zeta \in (0, 1]$ such that for all $g \in \mathbb{R}^n$, $\|g - P(g)\|_2^2 \leq (1 - \zeta^2) \|g\|_2^2$.

Assumption (E) is often referred to as the edge property and is quite standard in the literature [Grubb and Bagnell, 2011]. It measures the error of the approximated operator P on the direction of descent g_t.

A.2 Strongly convex function with smooth gradient

Theorem 4. *Let $(x_t)_t$ be a sequence generated by Algorithm 6. Assume that Assumptions (E), (SM) and (SC) hold. Let $\{x^\star\} = \arg\min_{x\in\mathbb{R}^n} F(x)$ (well defined by strong convexity), and choose $\lambda_t = \frac{\zeta^2}{8L}$. Then,*

$$F(x_T) - F(x^\star) \leq \left(1 - \frac{\zeta^4\kappa}{21L}\right)^T (F(x_0) - F(x^\star))\,.$$

Proof. First of all, let us remark that:

1. Assumption (SM) implies L-Lipschitz continuity of the gradient ∇F [Nesterov, 2004, Theorem 2.1.5]:
$$\forall x, x' \in \mathbb{R}^n, \qquad \|\nabla F(x) - \nabla F(x')\|_2 \leq L\,\|x - x'\|_2\,; \tag{5}$$
2. Assumption (SC) leads to the upper bound [Nesterov, 2004, Theorem 2.1.10]:
$$\forall x \in \mathbb{R}^n, \qquad 2\kappa\,(F(x) - F(x^\star)) \leq \|\nabla F(x)\|_2^2\,. \tag{6}$$

Then, from Assumption (SM) and by the update rule for x_{t+1} in Algorithm 6:

$$\begin{aligned} F(x_{t+1}) &\leq F(x_t) + \langle \nabla F(x_t), -\lambda_t P(g_t)\rangle + \frac{L\lambda_t^2}{2}\|P(g_t)\|_2^2 \\ &= F(x_t) - \lambda_t\langle g_t, P(g_t)\rangle - \lambda_t\langle \nabla F(x_t) - g_t, P(g_t)\rangle + \frac{L\lambda_t^2}{2}\|P(g_t)\|_2^2\,. \end{aligned} \tag{7}$$

Now, from Assumption (E):

$$\begin{aligned} -\lambda_t\langle g_t, P(g_t)\rangle &= \frac{\lambda_t}{2}\left(\|g_t - P(g_t)\|_2^2 - \|g_t\|_2^2 - \|P(g_t)\|_2^2\right) \\ &\leq \frac{\lambda_t}{2}\left[(1-\zeta^2)\,\|g_t\|_2^2 - \|g_t\|_2^2 - \|P(g_t)\|_2^2\right] \\ &= -\frac{\lambda_t\zeta^2}{2}\|g_t\|_2^2 - \frac{\lambda_t}{2}\|P(g_t)\|_2^2\,. \end{aligned} \tag{8}$$

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

$$\begin{aligned} \|\nabla F(x_t) - g_t\|_2 &= \left\|\nabla F(x_t) - \nabla F\left(\mathrm{prox}_{\lambda_t F}(x_t)\right)\right\|_2 \\ &\leq L\left\|x_t - \mathrm{prox}_{\lambda_t F}(x_t)\right\|_2 && \text{(Equation (5))} \\ &\leq \lambda_t L\,\|g_t\|_2 && \text{(definition of } g_t\text{).} \end{aligned} \tag{9}$$

So,

$$\begin{aligned} -\lambda_t\langle \nabla F(x_t) - g_t, P(g_t)\rangle &\leq \lambda_t\,\|\nabla F(x_t) - g_t\|_2\,\|P(g_t)\|_2 && \text{(Cauchy-Schwarz)} \\ &\leq \lambda_t^2 L\,\|g_t\|_2\,\|P(g_t)\|_2 && \text{(Equation (9))} \\ &\leq 2\lambda_t^2 L\,\|g_t\|_2^2\,, \end{aligned} \tag{10}$$

since $\|P(g_t)\|_2 \leq 2\,\|g_t\|_2$, by Assumption (E).

Combining Equations (7), (8) and (10):

$$\begin{aligned} F(x_{t+1}) &\leq F(x_t) - \frac{\lambda_t\zeta^2}{2}\|g_t\|_2^2 - \frac{\lambda_t}{2}\|P(g_t)\|_2^2 + 2\lambda_t^2 L\,\|g_t\|_2^2 + \frac{L\lambda_t^2}{2}\|P(g_t)\|_2^2 \\ &= F(x_t) - \lambda_t\left(\frac{\zeta^2}{2} - 2\lambda_t L\right)\|g_t\|_2^2 - \frac{\lambda_t}{2}\,(1 - L\lambda_t)\,\|P(g_t)\|_2^2\,. \end{aligned}$$

Now, choosing $\lambda_t = \frac{\zeta^2}{8L}$, one has $\lambda_t \left(\frac{\zeta^2}{2} - 2\lambda_t L\right) = \frac{\zeta^4}{32L}$ on one hand and $-\frac{\lambda_t}{2}(1 - L\lambda_t)\|P(g_t)\|_2^2 \leq 0$ on the other, leading to:

$$F(x_{t+1}) \leq F(x_t) - \frac{\zeta^4}{32L}\|g_t\|_2^2 . \tag{11}$$

Let us remark that, by Equation (6):

$$\begin{aligned}
2\kappa\left(F(x_t) - F(x^\star)\right) &\leq \|\nabla F(x_t)\|_2^2 \\
&\leq \left(\|\nabla F(x_t) - g_t\|_2 + \|g_t\|_2\right)^2 \\
&\leq (1 + \lambda_t L)^2 \|g_t\|_2^2 && \text{(Equation (9))} \\
&\leq \left(1 + \frac{\zeta^2}{8}\right)^2 \|g_t\|_2^2 && \left(\lambda_t = \frac{\zeta^2}{8L}\right),
\end{aligned}$$

that is,

$$\|g_t\|_2^2 \geq \frac{128\kappa}{(8+\zeta^2)^2}\left(F(x_t) - F(x^\star)\right).$$

So, from Equation (11),

$$\begin{aligned}
F(x_{t+1}) - F(x^\star) &\leq F(x_t) - F(x^\star) - \frac{\zeta^4}{32L}\|g_t\|_2^2 \\
&\leq \left(1 - \frac{\zeta^4}{32L}\frac{128\kappa}{(8+\zeta^2)^2}\right)\left(F(x_t) - F(x^\star)\right) \\
&= \left(1 - \frac{4\zeta^4}{(8+\zeta^2)^2}\frac{\kappa}{L}\right)\left(F(x_t) - F(x^\star)\right) \\
&\leq \left(1 - \frac{\zeta^4}{21}\frac{\kappa}{L}\right)\left(F(x_t) - F(x^\star)\right) && (\star) \\
&\leq \left(1 - \frac{\zeta^4\kappa}{21L}\right)^{t+1}\left(F(x_0) - F(x^\star)\right) && \text{(by induction)},
\end{aligned}$$

where we have used $(\star)$ that $\forall x \in [0,1]$, $\frac{4x^2}{(8+x)^2} \geq \frac{x^2}{21}$. □

A.3 Lipschitz continuous convex function

Lemma 5. *Let $(x_t)_t$ be a sequence generated by Algorithm 7. Assume that Assumptions (E), (SC) and (L) hold and that there exists $x^\star \in \arg\min_{x\in\mathbb{R}^n} F(x)$. Then,*

$$\begin{aligned}
&\min_{1\leq t\leq T} F(x_t) - F(x^\star) \\
&\leq \frac{1}{2T}\left(\frac{1}{\lambda_0} - \kappa\right)\|x_0 - x^\star\|_2^2 + \frac{1}{2T}\sum_{t=1}^{T-1}\left(\frac{1}{\lambda_t} - \frac{1}{\lambda_{t-1}} - \kappa\right)\|x_t - x^\star\|_2^2 \\
&+ \frac{1}{T}\sum_{t=0}^{T-1}\lambda_t\Bigg(\frac{1}{2}\|P(g_t + \Delta_t)\|_2^2 - \left(1 + \frac{\kappa\lambda_t}{2}\right)\|g_t\|_2^2 + \kappa\langle g_t, x_t - x^\star\rangle \\
&+ \langle\Delta_{t+1}, P(g_t + \Delta_t)\rangle + G\|g_t - P(g_t + \Delta_t)\|_2\Bigg) + \frac{\lambda_{T-1}}{2T}\|\Delta_T\|_2^2 .
\end{aligned}$$

In addition, the result still holds if $\kappa = 0$.

Lemma 5. For any non-negative integer $t < T$, let

$$y_{t+1} = x_t - \lambda_t g_t = \mathrm{prox}_{\lambda_t F}(x_t).$$

By construction, $g_t \in \partial F(y_{t+1})$, so

$$\begin{aligned}
F(x^\star) &\geq F(y_{t+1}) + \langle g_t, x^\star - y_{t+1}\rangle + \frac{\kappa}{2}\left\|y_{t+1} - x^\star\right\|_2^2 \\
&= F(y_{t+1}) + \langle g_t, x^\star - (x_t - \lambda_t g_t)\rangle + \frac{\kappa}{2}\left\|(x_t - \lambda_t g_t) - x^\star\right\|_2^2 \\
&= F(y_{t+1}) + \langle g_t, x^\star - x_t\rangle + \lambda_t \left\|g_t\right\|_2^2 + \frac{\kappa}{2}\left\|x_t - x^\star\right\|_2^2 + \frac{\kappa\lambda_t^2}{2}\left\|g_t\right\|_2^2 \\
&\quad - \kappa\lambda_t \left\langle g_t, x_t - x^\star\right\rangle \\
&= F(y_{t+1}) + \langle P(g_t + \Delta_t), x^\star - x_t\rangle + \left(\lambda_t + \frac{\kappa\lambda_t^2}{2}\right)\left\|g_t\right\|_2^2 + \frac{\kappa}{2}\left\|x_t - x^\star\right\|_2^2 \\
&\quad - \kappa\lambda_t \left\langle g_t, x_t - x^\star\right\rangle + \left\langle g_t - P(g_t + \Delta_t), x^\star - x_t\right\rangle . \qquad (12)
\end{aligned}$$

Now, let us analyze the potential $\left\|x_{t+1} - x^\star\right\|_2^2$:

$$\begin{aligned}
\left\|x_{t+1} - x^\star\right\|_2^2 &= \left\|x_t - \lambda_t P(g_t + \Delta_t) - x^\star\right\|_2^2 \\
&= \left\|x_t - x^\star\right\|_2^2 + \lambda_t^2 \left\|P(g_t + \Delta_t)\right\|_2^2 - 2\lambda_t \left\langle P(g_t + \Delta_t), x_t - x^\star\right\rangle .
\end{aligned}$$

Thus,

$$\left\langle P(g_t + \Delta_t), x_t - x^\star\right\rangle = \frac{1}{2\lambda_t}\left(\left\|x_t - x^\star\right\|_2^2 - \left\|x_{t+1} - x^\star\right\|_2^2\right) + \frac{\lambda_t}{2}\left\|P(g_t + \Delta_t)\right\|_2^2 . \qquad (13)$$

Combining Equation (12) and Equation (13), we obtain:

$$\begin{aligned}
F(y_{t+1}) - F(x^\star) &\leq \langle P(g_t + \Delta_t), x_t - x^\star\rangle - \left(\lambda_t + \frac{\kappa\lambda_t^2}{2}\right)\left\|g_t\right\|_2^2 - \frac{\kappa}{2}\left\|x_t - x^\star\right\|_2^2 \\
&\quad + \kappa\lambda_t \left\langle g_t, x_t - x^\star\right\rangle + \left\langle g_t - P(g_t + \Delta_t), x_t - x^\star\right\rangle \\
&\leq \frac{1}{2\lambda_t}\left(\left\|x_t - x^\star\right\|_2^2 - \left\|x_{t+1} - x^\star\right\|_2^2\right) - \frac{\kappa}{2}\left\|x_t - x^\star\right\|_2^2 \\
&\quad + \frac{\lambda_t}{2}\left\|P(g_t + \Delta_t)\right\|_2^2 - \left(\lambda_t + \frac{\kappa\lambda_t^2}{2}\right)\left\|g_t\right\|_2^2 \\
&\quad + \kappa\lambda_t \left\langle g_t, x_t - x^\star\right\rangle + \left\langle g_t - P(g_t + \Delta_t), x_t - x^\star\right\rangle . \qquad (14)
\end{aligned}$$

Now, remark that:

$$\begin{aligned}
&\sum_{t=0}^{T-1}\left(\left(\frac{1}{\lambda_t} - \kappa\right)\left\|x_t - x^\star\right\|_2^2 - \frac{1}{\lambda_t}\left\|x_{t+1} - x^\star\right\|_2^2\right) \\
&= \left(\frac{1}{\lambda_0} - \kappa\right)\left\|x_0 - x^\star\right\|_2^2 + \sum_{t=1}^{T-1}\left(\frac{1}{\lambda_t} - \kappa\right)\left\|x_t - x^\star\right\|_2^2 - \sum_{t=0}^{T-2}\frac{1}{\lambda_t}\left\|x_{t+1} - x^\star\right\|_2^2 \\
&\quad - \frac{1}{\lambda_{T-1}}\left\|x_T - x^\star\right\|_2^2 \\
&= \left(\frac{1}{\lambda_0} - \kappa\right)\left\|x_0 - x^\star\right\|_2^2 + \sum_{t=1}^{T-1}\left(\frac{1}{\lambda_t} - \frac{1}{\lambda_{t-1}} - \kappa\right)\left\|x_t - x^\star\right\|_2^2 \\
&\quad - \frac{1}{\lambda_{T-1}}\left\|x_T - x^\star\right\|_2^2 , \qquad (15)
\end{aligned}$$

and

$$\begin{aligned}
&\sum_{t=0}^{T-1} \langle g_t - P(g_t + \Delta_t), x_t - x^\star \rangle \\
&= \sum_{t=0}^{T-1} \langle g_t + \Delta_t - P(g_t + \Delta_t), x_{t+1} + \lambda_t P(g_t + \Delta_t) - x^\star \rangle - \sum_{t=0}^{T-1} \langle \Delta_t, x_t - x^\star \rangle \\
&= \sum_{t=0}^{T-1} \langle \Delta_{t+1}, x_{t+1} - x^\star \rangle - \sum_{t=0}^{T-1} \langle \Delta_t, x_t - x^\star \rangle + \sum_{t=0}^{T-1} \lambda_t \langle \Delta_{t+1}, P(g_t + \Delta_t) \rangle \\
&= \langle \Delta_T, x_T - x^\star \rangle - \langle \Delta_0, x_0 - x^\star \rangle + \sum_{t=0}^{T-1} \lambda_t \langle \Delta_{t+1}, P(g_t + \Delta_t) \rangle \\
&= \langle \Delta_T, x_T - x^\star \rangle + \sum_{t=0}^{T-1} \lambda_t \langle \Delta_{t+1}, P(g_t + \Delta_t) \rangle \,, \qquad (16)
\end{aligned}$$

since $\Delta_0 = 0$.

Then, by summation of Equation (14) and using Equation (15),

$$\begin{aligned}
&\sum_{t=0}^{T-1} (F(y_{t+1}) - F(x^\star)) \\
&\leq \frac{1}{2} \sum_{t=0}^{T-1} \left(\left(\frac{1}{\lambda_t} - \kappa \right) \|x_t - x^\star\|_2^2 - \frac{1}{\lambda_t} \|x_{t+1} - x^\star\|_2^2 \right) \\
&+ \sum_{t=0}^{T-1} \lambda_t \left(\frac{1}{2} \|P(g_t + \Delta_t)\|_2^2 - \left(1 + \frac{\kappa \lambda_t}{2} \right) \|g_t\|_2^2 + \kappa \langle g_t, x_t - x^\star \rangle \right) \\
&+ \sum_{t=0}^{T-1} \langle g_t - P(g_t + \Delta_t), x_t - x^\star \rangle \\
&= \left(\frac{1}{2\lambda_0} - \frac{\kappa}{2} \right) \|x_0 - x^\star\|_2^2 + \frac{1}{2} \sum_{t=1}^{T-1} \left(\frac{1}{\lambda_t} - \frac{1}{\lambda_{t-1}} - \kappa \right) \|x_t - x^\star\|_2^2 \\
&+ \sum_{t=0}^{T-1} \lambda_t \left(\frac{1}{2} \|P(g_t + \Delta_t)\|_2^2 - \left(1 + \frac{\kappa \lambda_t}{2} \right) \|g_t\|_2^2 + \kappa \langle g_t, x_t - x^\star \rangle \right) \\
&+ \sum_{t=0}^{T-1} \langle g_t - P(g_t + \Delta_t), x_t - x^\star \rangle - \frac{1}{2\lambda_{T-1}} \|x_T - x^\star\|_2^2 \,.
\end{aligned}$$

Now, using Equation (16),

$$\begin{aligned}
&\sum_{t=0}^{T-1}(F(y_{t+1})-F(x^\star))\\
&\leq \left(\frac{1}{2\lambda_0}-\frac{\kappa}{2}\right)\|x_0-x^\star\|_2^2+\frac{1}{2}\sum_{t=1}^{T-1}\left(\frac{1}{\lambda_t}-\frac{1}{\lambda_{t-1}}-\kappa\right)\|x_t-x^\star\|_2^2\\
&+\sum_{t=0}^{T-1}\lambda_t\Bigg(\frac{1}{2}\|P(g_t+\Delta_t)\|_2^2-\left(1+\frac{\kappa\lambda_t}{2}\right)\|g_t\|_2^2+\kappa\langle g_t,x_t-x^\star\rangle\\
&+\langle\Delta_{t+1},P(g_t+\Delta_t)\rangle\Bigg)+\langle\Delta_T,x_T-x^\star\rangle-\frac{1}{2\lambda_{T-1}}\|x_T-x^\star\|_2^2\\
&\leq \left(\frac{1}{2\lambda_0}-\frac{\kappa}{2}\right)\|x_0-x^\star\|_2^2+\frac{1}{2}\sum_{t=1}^{T-1}\left(\frac{1}{\lambda_t}-\frac{1}{\lambda_{t-1}}-\kappa\right)\|x_t-x^\star\|_2^2\\
&+\sum_{t=0}^{T-1}\lambda_t\Bigg(\frac{1}{2}\|P(g_t+\Delta_t)\|_2^2-\left(1+\frac{\kappa\lambda_t}{2}\right)\|g_t\|_2^2+\kappa\langle g_t,x_t-x^\star\rangle\\
&+\langle\Delta_{t+1},P(g_t+\Delta_t)\rangle\Bigg)+\frac{\lambda_{T-1}}{2}\|\Delta_T\|_2^2,
\end{aligned}$$

where the last line comes from $bx-ax^2\leq\frac{b^2}{4a}$ for any $a>0$ and $b\in\mathbb{R}$.

To conclude,

$$\begin{aligned}
&\min_{1\leq t\leq T}F(x_t)-F(x^\star)\\
&\leq\frac{1}{T}\sum_{t=0}^{T-1}(F(y_{t+1})-F(x^\star))+\frac{1}{T}\sum_{t=0}^{T-1}(F(x_{t+1})-F(y_{t+1}))\\
&\leq\frac{1}{T}\sum_{t=0}^{T-1}(F(y_{t+1})-F(x^\star))+\frac{1}{T}\sum_{t=0}^{T-1}G\|(x_t-\lambda_tP(g_t+\Delta_t))-(x_t-\lambda_tg_t)\|_2\\
&\leq\frac{1}{2T}\left(\frac{1}{\lambda_0}-\kappa\right)\|x_0-x^\star\|_2^2+\frac{1}{2T}\sum_{t=1}^{T-1}\left(\frac{1}{\lambda_t}-\frac{1}{\lambda_{t-1}}-\kappa\right)\|x_t-x^\star\|_2^2\\
&+\frac{1}{T}\sum_{t=0}^{T-1}\lambda_t\Bigg(\frac{1}{2}\|P(g_t+\Delta_t)\|_2^2-\left(1+\frac{\kappa\lambda_t}{2}\right)\|g_t\|_2^2+\kappa\langle g_t,x_t-x^\star\rangle\\
&+\langle\Delta_{t+1},P(g_t+\Delta_t)\rangle+G\|g_t-P(g_t+\Delta_t)\|_2\Bigg)+\frac{\lambda_{T-1}}{2T}\|\Delta_T\|_2^2.
\end{aligned}$$

□

Theorem 6. *Let $(x_t)_t$ be a sequence generated by Algorithm 7. Assume that Assumptions (E) and (L) hold. Assume also that there exists a minimizer $x^\star\in\arg\min_{x\in\mathbb{R}^n}F(x)$ and that $\|x_t\|_2\leq R$ and $\|x^\star\|_2\leq R$ (for some $R>0$ and all t). Then, choosing $\lambda_t=\frac{1}{\sqrt{t+1}}$ leads to:*

$$\min_{1\leq t\leq T}F(x_t)-F(x^\star)\leq\frac{2R^2}{\sqrt{T}}+\frac{2G^2}{\zeta^4\sqrt{T}}\left(20+\frac{1}{T}\right).$$

Proof. By Lemma 5 with $\kappa = 0$ and $\lambda_t = \frac{1}{\sqrt{t+1}}$, we have:

$$\begin{aligned}
\min_{1\le t\le T} F(x_t) - F(x^\star) &\le \frac{1}{2\lambda_0 T}\left\|x_0 - x^\star\right\|_2^2 + \frac{1}{2T}\sum_{t=1}^{T-1}\left(\frac{1}{\lambda_t} - \frac{1}{\lambda_{t-1}}\right)\left\|x_t - x^\star\right\|_2^2 \\
&\quad + \frac{1}{T}\sum_{t=0}^{T-1}\lambda_t\Bigg(\frac{1}{2}\left\|P(g_t+\Delta_t)\right\|_2^2 - \left\|g_t\right\|_2^2 \\
&\quad + \left\langle \Delta_{t+1}, P(g_t+\Delta_t)\right\rangle + G\left\|g_t - P(g_t+\Delta_t)\right\|_2\Bigg) \\
&\quad + \frac{\lambda_{T-1}}{2T}\left\|\Delta_T\right\|_2^2 \\
&\le \frac{1}{2T}\sum_{t=0}^{T-1}\left(\sqrt{t+1} - \sqrt{t}\right)\left\|x_t - x^\star\right\|_2^2 \\
&\quad + \frac{1}{T}\sum_{t=0}^{T-1}\frac{1}{\sqrt{t+1}}\Bigg(\frac{1}{2}\left\|P(g_t+\Delta_t)\right\|_2^2 + \left\langle \Delta_{t+1}, P(g_t+\Delta_t)\right\rangle \\
&\quad + G\left\|g_t - P(g_t+\Delta_t)\right\|_2\Bigg) + \frac{1}{2T^{\frac{3}{2}}}\left\|\Delta_T\right\|_2^2 \\
&\le \frac{2R^2}{\sqrt{T}} + \frac{1}{T}\sum_{t=0}^{T-1}\frac{1}{\sqrt{t+1}}\Bigg(\frac{1}{2}\left\|P(g_t+\Delta_t)\right\|_2^2 \\
&\quad + \left\langle \Delta_{t+1}, P(g_t+\Delta_t)\right\rangle + G\left\|g_t - P(g_t+\Delta_t)\right\|_2\Bigg) \\
&\quad + \frac{1}{2T^{\frac{3}{2}}}\left\|\Delta_T\right\|_2^2.
\end{aligned}$$

Now, since $g_t \in \partial F(y_{t+1})$, $\left\|g_t\right\|_2 \le G$. In addition, since $\Delta_0 = 0$, by Assumption (E),

$$\begin{aligned}
\left\|\Delta_{T+1}\right\|_2 &\le \sqrt{1-\zeta^2}\left\|g_T + \Delta_T\right\|_2 \\
&\le \sqrt{1-\zeta^2}\left\|g_T\right\|_2 + \sqrt{1-\zeta^2}\left\|\Delta_T\right\|_2 \\
&\le \sum_{t=0}^{T}\sqrt{1-\zeta^2}^{T+1-t}\left\|g_t\right\|_2 \\
&\le G\sum_{t=1}^{T+1}\sqrt{1-\zeta^2}^{t} \\
&\le \frac{\sqrt{1-\zeta^2}}{1-\sqrt{1-\zeta^2}}G \\
&\le \frac{2}{\zeta^2}G,
\end{aligned}$$

where we have used that $\frac{1}{1-\sqrt{1-\zeta^2}} \leq \frac{2}{\zeta^2}$. Moreover,

$$\begin{aligned}
\|P(g_t+\Delta_t)\|_2 &\leq \|P(g_t+\Delta_t)-(g_t+\Delta_t)\|_2 + \|g_t+\Delta_t\|_2 \\
&\leq (\sqrt{1-\zeta^2}+1)\,\|g_t+\Delta_t\|_2 \\
&\leq (\sqrt{1-\zeta^2}+1)G + (\sqrt{1-\zeta^2}+1)\frac{\sqrt{1-\zeta^2}}{1-\sqrt{1-\zeta^2}}G \\
&\leq \frac{(1-(1-\zeta^2))+(\sqrt{1-\zeta^2}+1)\sqrt{1-\zeta^2}}{1-\sqrt{1-\zeta^2}}G \\
&\leq \frac{\zeta^2+(1-\zeta^2)+\sqrt{1-\zeta^2}}{1-\sqrt{1-\zeta^2}}G \\
&\leq \frac{1+\sqrt{1-\zeta^2}}{1-\sqrt{1-\zeta^2}}G \\
&\leq \frac{4}{\zeta^2}G.
\end{aligned}$$

At last,

$$\begin{aligned}
\|g_t - P(g_t+\Delta_t)\|_2 &\leq \|g_t+\Delta_t-P(g_t+\Delta_t)\|_2 + \|\Delta_t\|_2 \\
&\leq \sqrt{1-\zeta^2}\,\|g_t+\Delta_t\|_2 + \|\Delta_t\|_2 \\
&\leq \sqrt{1-\zeta^2}\,\|g_t\|_2 + (\sqrt{1-\zeta^2}+1)\,\|\Delta_t\|_2 \\
&\leq \sqrt{1-\zeta^2}G + (\sqrt{1-\zeta^2}+1)\frac{\sqrt{1-\zeta^2}}{1-\sqrt{1-\zeta^2}}G \\
&\leq \frac{(\sqrt{1-\zeta^2}-(1-\zeta^2))+(1-\zeta^2+\sqrt{1-\zeta^2})}{1-\sqrt{1-\zeta^2}}G \\
&\leq \frac{2\sqrt{1-\zeta^2}}{1-\sqrt{1-\zeta^2}}G \\
&\leq \frac{4}{\zeta^2}G.
\end{aligned}$$

To conclude,

$$\begin{aligned}
\min_{1\leq t\leq T} F(x_t) - F(x^\star) &\leq \frac{2R^2}{\sqrt{T}} \\
&\quad + \frac{1}{T}\sum_{t=0}^{T-1}\frac{1}{\sqrt{t+1}}\left(\frac{1}{2}\left(\frac{4}{\zeta^2}G\right)^2 + \frac{2}{\zeta^2}G\frac{4}{\zeta^2}G + \frac{4}{\zeta^2}G^2\right) \\
&\quad + \frac{1}{2T^{\frac{3}{2}}}\frac{4}{\zeta^4}G^2 \\
&\leq \frac{2R^2}{\sqrt{T}} + \frac{2G^2}{\zeta^4\sqrt{T}}\left(20+\frac{1}{T}\right),
\end{aligned}$$

where we have used that $\sum_{t=0}^{T-1}\frac{1}{\sqrt{t+1}} \leq 2\sqrt{T}$ and $\frac{1}{\zeta^2} \leq \frac{1}{\zeta^4}$. □

B Implementation details

This section provides detailed calculations for each step of Algorithm 5 applied to Problem (P1) and for all losses presented in Table 2.

It is possible that a step has no closed-form expression but is the root of an equation. In this case (which is indicated by * below), the Newton-Raphson iteration is provided. In practice, less than 10 iterations of the Newton-Raphson method are enough to obtain a good approximation.

For now on, let us note, for all $x \in \mathbb{R}$,

$$\operatorname{sign}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases}$$

B.1 Least squares loss

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

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

Subgradient: $\widetilde{\nabla}_n C(f_t) = \left(\frac{f_t(X_i)-Y_i}{n}\right)_{1\le i\le n}$.

Proximal direction: $\operatorname{Prox}_n^\lambda C(f_t) = \left(\frac{f_t(X_i)-Y_i}{\lambda+n}\right)_{1\le i\le n}$.

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

B.2 Least absolute deviations loss

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

Initial estimator: f_0 is the empirical median of the sample $\{Y_1, \dots, Y_n\}$.

Subradient: $\widetilde{\nabla}_n C(f_t) = \left(\frac{\operatorname{sign}(f_t(X_i)-Y_i)}{n}\right)_{1\le i\le n}$.

Proximal direction:

$$\begin{aligned} \operatorname{Prox}_n^\lambda C(f_t) &= \left(\frac{f_t(X_i) - Y_i}{\max\left(\lambda, n|f_t(X_i) - Y_i|\right)}\right)_{1\le i\le n} \\ &= \left(\begin{cases} \frac{\operatorname{sign}(f_t(X_i)-Y_i)}{n} & \text{if } |f_t(X_i) - Y_i| > \frac{\lambda}{n} \\ \frac{f_t(X_i)-Y_i}{\lambda} & \text{otherwise} \end{cases}\right)_{1\le i\le n}. \end{aligned}$$

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

B.3 Pinball loss

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 $\{Y_1, \dots, Y_n\}$.

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

Proximal direction:

$$\mathrm{Prox}_n^\lambda C(f_t) = \left(\begin{cases} -\frac{\tau}{n} & \text{if } Y_i - f_t(X_i) > \frac{\lambda\tau}{n} \\ \frac{1-\tau}{n} & \text{if } Y_i - f_t(X_i) < \frac{\lambda(\tau-1)}{n} \\ \frac{f_t(X_i)-Y_i}{\lambda} & \text{otherwise} \end{cases} \right)_{1 \le i \le n}.$$

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

B.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 $p = \sum_{\substack{1 \le i \le n \\ Y_i = 1}} 1$.

Subgradient: $\widetilde{\nabla}_n C(f_t) = \left(\frac{-\beta Y_i e^{-Y_i f_t(X_i)}}{n} \right)_{1 \le i \le n}$.

Proximal direction*: $\mathrm{Prox}_n^\lambda C(f_t) = \left(\frac{f_t(X_i) - u_i}{\lambda} \right)_{1 \le i \le n}$, with Newton-Raphson iteration $u_i \leftarrow u_i + \frac{f_t(X_i) - u_i + \frac{\lambda\beta Y_i}{n} \mathrm{e}^{-\beta Y_i u_i}}{1 + \frac{\lambda\beta^2}{n} \mathrm{e}^{-\beta Y_i u_i}}$.

Line search*: Newton-Raphson iteration

$$\gamma_{t+1} \leftarrow \gamma_{t+1} + \frac{\sum_{i=1}^n Y_i g_{t+1}(X_i) \, \mathrm{e}^{-\beta Y_i (f_t(X_i) + \gamma_{t+1} g_{t+1}(X_i))}}{\beta \sum_{i=1}^n g_{t+1}(X_i)^2 \, \mathrm{e}^{-\beta Y_i (f_t(X_i) + \gamma_{t+1} g_{t+1}(X_i))}}.$$

B.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 $p = \sum_{\substack{1 \le i \le n \\ Y_i = 1}} 1$.

Subgradient: $\widetilde{\nabla}_n C(f_t) = \left(\frac{-Y_i e^{-Y_i f_t(X_i)}}{n \log 2 (1 + e^{-Y_i f_t(X_i)})} \right)_{1 \le i \le n}$.

Proximal direction*: $\mathrm{Prox}_n^\lambda C(f_t) = \left(\frac{f_t(X_i) - u_i}{\lambda} \right)_{1 \le i \le n}$, with Newton-Raphson iteration $u_i \leftarrow u_i + \frac{f_t(X_i) - u_i + \frac{\lambda}{n \log 2} \frac{Y_i \mathrm{e}^{-Y_i u_i}}{1 + \mathrm{e}^{-Y_i u_i}}}{1 + \frac{\lambda}{n \log 2} \frac{\mathrm{e}^{-Y_i u_i}}{\left(1 + \mathrm{e}^{-Y_i u_i}\right)^2}}$.

Line search*: Newton-Raphson iteration

$$\gamma_{t+1} \leftarrow \gamma_{t+1} + \frac{\sum_{i=1}^n \frac{Y_i g_{t+1}(X_i) \, \mathrm{e}^{-Y_i (f_t(X_i) + \gamma_{t+1} g_{t+1}(X_i))}}{1 + \mathrm{e}^{-Y_i (f_t(X_i) + \gamma_{t+1} g_{t+1}(X_i))}}}{\sum_{i=1}^n \frac{g_{t+1}(X_i)^2 \, \mathrm{e}^{-Y_i (f_t(X_i) + \gamma_{t+1} g_{t+1}(X_i))}}{\left(1 + \mathrm{e}^{-Y_i (f_t(X_i) + \gamma_{t+1} g_{t+1}(X_i))}\right)^2}}.$$

B.6 Hinge loss

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

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

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

Proximal direction:

$$\operatorname{Prox}_n^{\lambda} C(f_t) = \left(\begin{cases} -\frac{Y_i}{n} & \text{if } Y_i f_t(X_i) < 1 - \frac{\lambda}{n} \\ 0 & \text{if } Y_i f_t(X_i) > 1 \\ \frac{f_t(X_i) - Y_i}{\lambda} & \text{otherwise} \end{cases} \right)_{1 \leq i \leq n}.$$

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

C Accelerated proximal boosting in practice

Algorithm 8 describes a practical version of accelerated proximal boosting (Algorithm 4), which holds true also for accelerated gradient boosting [Biau et al., 2019]. In accordance with the practice, the proximal steps are chosen adaptively by a line search (Line 8 of Algorithm 8) and a shrinkage coefficient is introduced.

Algorithm 8 Accelerated proximal/gradient boosting in practice.

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 (see Appendix B)

$$\begin{cases} r \leftarrow -\widetilde{\nabla}_n C(f_t) & \text{for gradient boosting,} \\ r \leftarrow -\operatorname{Prox}_n^{\lambda} C(f_t) & \text{for proximal boosting.} \end{cases}$$

7: Compute $g_{t+1} \in \arg\min_{g \in \mathcal{F}} \|g(X_1^n) - r\|_2$.
8: Compute $\gamma_{t+1} \in \arg\min_{\gamma \in \mathbb{R}} C(f_t + \gamma g_{t+1})$ (see Appendix B).
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: Update weights $(w_0^{(t+1)}, \ldots, w_{t+1}^{(t+1)})$ according to Property 8.
12: **end for**

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

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 w_t: $f_T = \sum_{t=0}^{T} w_t g_t$. For this purpose, the weights of the final model have to be tracked despite the recursive update of f_{t+1} (Line 5 in Algorithm 4 and Line 9 in Algorithm 8):

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

Property 7 gives the closed-form expression of the weights of f_T in this case.

Property 7. *The weights of f_T are:*

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

Proof. The update rule in Line 5 in Algorithm 4 is:

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

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:

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

It follows that:

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

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:

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

□

In addition, Property 8 provides a recursive update suitable for implementing Algorithm 8. Let us remark that, Property 8 is also valid for accelerated gradient boosting as proposed by Biau et al. [2019]. This paves the way to efficient implementations of both accelerated proximal and accelerated gradient boosting, as done in the Python package `optboosing`[2].

Property 8. *Let $f_t = \sum_{j=0}^{t} w_j^{(t)} g_j$ be the expansion of f_t at iteration $t \in \{1, \dots, T-1\}$. Then, the weights can be updated according to the following recursion:*

$$\begin{cases} w_0^{(0)} = 1 \\ w_1^{(0)} = \nu\gamma_1 \\ w_1^{(1)} = \nu\gamma_1 \\ w_j^{(t+1)} = (w_j^{(t)} - w_j^{(t-1)})(1 + \alpha_t) + w_j^{(t-1)}, \forall j \in \{1, \dots, t\} \\ w_{t+1}^{(t+1)} = \nu\gamma_{t+1}. \end{cases} \tag{17}$$

Proof. See proof of Property 7. □

[2] https://github.com/msangnier/optboosting