Proximal boosting and variants

Erwan Fouillen, Claire Boyer, Maxime Sangnier

Proximal boosting and variants

Erwan Fouillen, Claire Boyer, and Maxime Sangnier

Sorbonne Université, CNRS, LPSM, Paris, France

July 27, 2021

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. Besides being motivated by non-differentiable optimization, the proposed algorithm benefits from algorithmic improvements such as controlling the approximation error and Nesterov's acceleration, in the same way as gradient boosting [Grubb and Bagnell, 2011, Biau et al., 2018]. This leads to two variants, respectively called residual proximal boosting and accelerated proximal boosting. Theoretical convergence is proved for the first 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 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 in data challenges.

As explained by Biau et al. [2018], gradient boosting has its roots in Freund and Schapire's work on combining classifiers, which resulted in the Adaboost algorithm [Schapire, 1990, Freund, 1995, Freund and Schapire, 1996, 1997]. Later, Friedman and colleagues developed a novel boosting procedure inspired by the numerical optimization literature, and nicknamed gradient boosting [Friedman et al., 2000, Friedman, 2001, 2002]. Such a connection of boosting between statistics and optimization was already stated in several previous analyses by Breiman [Breiman, 1997, 1998, 1999, 2000, 2004] and reviewed as functional optimization [Mason et al., 2000b,a, Meir and Rätsch, 2003, Bühlmann and

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

In gradient boosting (and variants), the number of weak learners 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., 2018]. While all these variants rely on differentiable loss functions, Grubb and Bagnell [2011] discuss the limitations of boosting with gradient descent in the non-differentiable setting, and tackle these issues by proposing two modified versions of (sub)gradient boosting, 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 a procedure to efficiently learn boosted models with non-differentiable loss functions, that can benefit from advantages of controlling the approximation error and accelerating the optimization procedure (respectively like residual gradient boosting [Grubb and Bagnell, 2011] and accelerated gradient boosting [Biau et al., 2018]).

To go into details, Section 2 reviews boosting with respect to the empirical risk minimization principle and illustrates the flaw of the current learning procedure in a simple non-differentiable case: least absolute deviations. Then, some backgrounds on non-smooth optimization are stated in Section 3 and we explain the main contribution of this paper: adapting the proximal point algorithm [Nesterov, 2004] to boosting. The proposed method is nicknamed proximal boosting and comes with two variants, residual proximal boosting and accelerated proximal boosting inspired by the developments mentioned above. 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). Regarding accelerated proximal boosting, we empirically observe that it may diverge but always in the overfitting regime, which is not harmful per se from a statistical point of view. Finally, the numerical study described in Section 5 shines a light on advantages and limitations of the proposed boosting procedures.

2 Problem and notation

Let $\mathcal{X}$ be an arbitrary input space and $\mathcal{Y} \subseteq \mathbb{R}$ an output space. Given a pair of random variables $(X, Y) \in \mathcal{X} \times \mathcal{Y}$, supervised learning aims at explaining Y given X, thanks to a 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

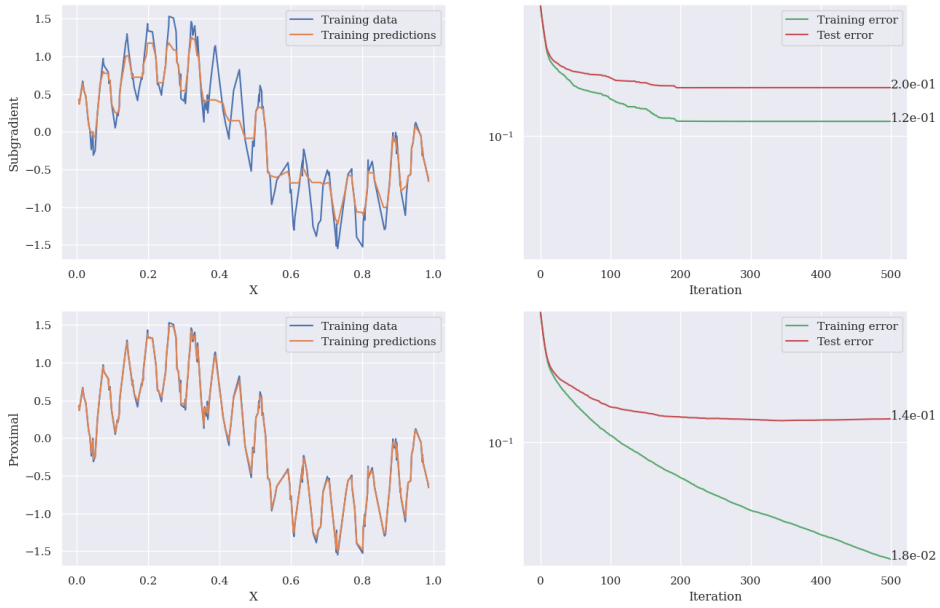

Figure 1: Predicted values and training error of a boosting machine trained with a subgradient (top) and a proximal-based method (bottom).

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 $\operatorname{span} \mathcal{F} = \{\sum_{t=1}^{m} w_t g_t : w \in \mathbb{R}^m, (g_1, \ldots, g_m) \in \mathcal{F}^m, m \in \mathbb{N}\}$ 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 $Y = \sin(2\pi X) + \epsilon$, 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 300): a traditional one with a subgradient-type method (Algorithm 1), and another with the proposed proximal-based procedure (Algorithm 2). Fig. 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).

In an optimization perspective, it appears clearly that the subgradient method fails to minimize the empirical risk (prediction is far from the data and the training error attains a plateau at $2 \cdot 10^{-1}$) 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 prevails 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 Fig. 1).

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 $\nabla_{\mathrm{n}} C(f) = \nabla D(f(X_1^n))$. On the contrary, if ℓ is not differentiable, we may consider a subgradient of C at f, denoted $\widetilde{\nabla}_{\mathrm{n}} C(f)$ and defined as any subgradient of D at $f(X_1^n)$.

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

3.1 Accelerated proximal gradient method

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

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

which is well defined by convexity and lower semi-continuity of h [Combettes and Wajs, 2005]. Then, the iterative procedure defined by choosing any $x_0 = v_0 \in \mathbb{R}^d$ and by setting for all $t \in \mathbb{N}$:

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

where $\gamma_{t+1} \in (0, 2/L)$ and $(\alpha_t)_t$ will be made precise thereafter, is known to converge to a minimizer of $g + h$ [Nesterov, 2004]. The rate of convergence depends on the choice of α_t: if $\alpha_t = 0$ for all $t \in \mathbb{N}$, then the previous procedure leads to the well known proximal gradient method, which converges in $O(1/t)$. More formally, assuming that $g+h$ has a minimizer $x^\star$, then $(g+h)(x_t) - (g+h)(x^\star) = O(1/t)$. On the other hand, if one chooses the sequence $(\alpha_t)_t$ defined recursively by:

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

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

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

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

- the proximal point algorithm ($g = 0$):
$$x_{t+1} = \operatorname{prox}_{\gamma_{t+1} h}(v_t) = v_t - \gamma_{t+1} \left[\frac{1}{\gamma_{t+1}} \left(v_t - \operatorname{prox}_{\gamma_{t+1} h}(v_t) \right) \right], \tag{2}$$
minimizes a single function h, which is only required to be convex and lower semi-continuous (in that case, there is no restriction on the step size γ_{t+1}, except being positive).

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

$$x_{t+1} = x_t - \gamma_{t+1} \left[\frac{1}{\gamma_{t+1}} \left(x_t - \operatorname{prox}_{\gamma_{t+1} h}(x_t) \right) \right]. \tag{3}$$

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

$$x_{t+1} = x_t - \gamma_{t+1} \widetilde{\nabla} h(x_t), \tag{4}$$

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 optimal, meaning that it cannot be improved without extra assumptions on h [Nesterov, 2004, Theorem 3.2.1]. Consequently, there does not exist an acceleration scheme for this class of objective functions.

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 (3)) or the accelerated proximal point method (described in Equation (2)). This motivation is emphasized by the fact that moving from subgradient to proximal point method only requires to replace the update direction $\widetilde{\nabla} h(x_t)$ by $\frac{1}{\gamma_{t+1}}(x_t - \operatorname{prox}_{\gamma_{t+1} h}(x_t))$. This observation is the cornerstone of the 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 [Mason et al., 2000a, Friedman, 2001]. 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 $w_0 = 1$.

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 C.

Algorithm 1 Gradient boosting.

Input: $\gamma_1, \dots, \gamma_T > 0$ (gradient steps).

Set $f_0 \in \arg\min_{g \in \mathcal{F}_0} C(g)$ (initialization).
for $t = 0$ **to** $T - 1$ **do**
  Compute $r \leftarrow -\widetilde{\nabla}_{\mathrm{n}} C(f_t)$ (pseudo-residuals).
  Compute $g_{t+1} \in \arg\min_{g \in \mathcal{F}} \|g(X_1^n) - r\|_{\ell_2}$.
  Set $f_{t+1} \leftarrow f_t + \gamma_{t+1} g_{t+1}$. (update).
end for

Output: f_T.

3.3 Boosting with non-differentiable loss functions

When the function ℓ is not differentiable with respect to its second argument, gradient boosting just uses a subgradient $\widetilde{\nabla}_{\mathrm{n}} C(f_t)$ instead of the gradient $\nabla_{\mathrm{n}} C(f_t)$. This is, of course, convenient but as explained previously, far from leading to interesting convergence 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 are the update directions of the optimization variable, which are respectively $\widetilde{\nabla}_{\mathrm{n}} C(f_t)$ and $\mathrm{Prox}_n^{\lambda_{t+1}} C(f_t)$, 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 $\widetilde{\nabla}_{\mathrm{n}} C(f_t)$ and leaves the rest unchanged, as described in Algorithm 2.

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 $r \leftarrow -\mathrm{Prox}_n^{\lambda_{t+1}} C(f_t)$ (pseudo-residuals).
  Compute $g_{t+1} \in \arg\min_{g \in \mathcal{F}} \|g(X_1^n) - r\|_{\ell_2}$.
  Set $f_{t+1} \leftarrow f_t + \lambda_{t+1} g_{t+1}$.
end for

Output: f_T.

We now introduce two alterations of Algorithm 2. As we will see in the next section, convergence of Algorithm 2 may not be guaranteed with weak assumptions on the function C. For this reason, the first variant of Algorithm 2, named residual proximal boosting and described in Algorithm 3, incorporates a mechanism introduced by Grubb and Bagnell [2011] in order to control the approximation error made at each iteration. In practice, it consists in augmenting the pseudo-residuals with the approximation error Δ_t of the previous iteration, which turns out to guarantee convergence in a quite general situation.

The second variant of Algorithm 2, named accelerated proximal boosting and described in Algorithm 4, consists in incorporating Nesterov's acceleration (as reviewed in Section 3.1) 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 uses the auxiliary function h_t instead of f_t in order to compute f_{t+1}. This also results in an estimator $f_T = \sum_{t=0}^{T} w_t g_t$ returned by Algorithm 4, where the weights $(w_0, \dots, w_T)$ are now given by a recursive formula (see Appendix C).

The idea at the core of Algorithm 4 originates from Biau et al. [2018], which adapts Nesterov's acceleration to gradient boosting. Algorithm 4 goes a step further by providing an accelerated boosting scheme for non-differentiable functions. Conceptually, Algorithm 4 seems more grounded than applying accelerated gradient boosting [Biau et al., 2018] to non-differentiable loss functions since the optimization theory predicts $O(1/t^2)$ convergence for the proximal point method but the inability for

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

the subgradient algorithm to be accelerated (see Section 3.1).

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 $g_{t+1} \in \arg\min_{g \in \mathcal{F}} \|g(X_1^n) - r\|_{\ell_2}$.
  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 proximal boosting (presented in Algorithm 2) and residual proximal boosting (Algorithm 3). Algorithm 4 is not addressed here for it will be seen that it may diverge sometimes (even though it is not a real flaw in practice since divergence occurs in the overfitting regime).

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:*

$$\varepsilon_{t+1} = \left\| g_{t+1}(X_1^n) + \operatorname{Prox}_n^{\lambda_{t+1}} C(f_t) \right\|_{\ell_2}.$$

Suppose that $\{f(X_1^n)_t\}_t$ is bounded and that

$$\sum_{t=0}^{+\infty} \varepsilon_t < +\infty. \tag{5}$$

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:

$$\forall r \in \mathbb{R}^n, \qquad \exists g \in \mathcal{F} : \|g(X_1^n) - r\|_{\ell_2}^2 \leq (1 - \zeta^2) \|r\|_{\ell_2}^2 .$$

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 assumptions similar to that coming up in the theoretical analysis of gradient boosting proposed by [Grubb and Bagnell, 2011]: 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 $x, x' \in \mathbb{R}^n$,

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

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

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

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

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

Proof. Given that $\forall f \in \mathscr{F}_{\mathcal{X}} : C(f) = D(f(X_1^n))$, this is a straightforward application of Theorem 4 to the function D. □

Theorem 2 states that proximal boosting has a linear convergence rate under smoothness and strong convexity assumptions. This result is similar to that obtained for gradient boosting [Grubb and Bagnell, 2011] and is indeed based on the same assumptions.

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, which incorporates a mechanism introduced by Grubb and Bagnell [2011], aimed at preventing the approximation error from diverging (as in Equation (5)).

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 $x, x' \in \mathbb{R}^n$,

$$|D(x) - D(x')| \leq G \|x - x'\|_{\ell_2} .$$

A convergence rate for residual proximal boosting (Algorithm 3) can be derived from this property, as stated in Theorem 3 (which is based on a result presented 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 $\|f_t(X_1^n)\|_{\ell_2} \leq R$ and $\|f^\star(X_1^n)\|_{\ell_2} \leq R$ 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 $\forall f \in \mathscr{F}_{\mathcal{X}} : C(f) = D(f(X_1^n))$, this is a straightforward application of Theorem 6 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. This is perfectly similar to residual subgradient boosting, as introduced by Grubb and Bagnell [2011]. It may be disappointing to obtain similar rates while Algorithm 3 is expected to better handle the non-differentiability of the objective function by using the proximal operator instead of any subgradient. However, the approximation made by a weak learner introduces an error that boils down to be similar to that made in residual subgradient boosting [Grubb and Bagnell, 2011].

In practice, Section 5 will show that linear convergence (as stated by Theorem 2) is often observed in numerical applications, even though the loss function is not differentiable.

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.

Algorithm 5 Meta-algorithm for proximal boosting.

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

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

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

3: Compute pseudo-residuals $r \in \mathbb{R}^n$ (depending on λ).

4: Compute $g_{t+1} \in \arg\min_{g \in \mathcal{F}} \|g(X_1^n) - (r + \Delta_t)\|_{\ell_2}$.

5: Set

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

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

7: Set

$$\begin{cases} f_{t+1} \leftarrow f_t + \alpha_t(f_t - f_{t-1}) + \nu\gamma_{t+1} g_{t+1} & \text{for accelerated boosting,} \\ f_{t+1} \leftarrow f_t + \nu\gamma_{t+1} g_{t+1} & \text{otherwise.} \end{cases}$$

8: **end for**

Output: f_T.

First of all, the proximal step is fixed to some value $\lambda > 0$: $\lambda_t = \lambda$, and the update rule (let us take proximal boosting as an example) $f_{t+1} \leftarrow f_t + \lambda_{t+1} g_{t+1}$ 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, as 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.

At last, for a more detailed description of accelerated proximal (or gradient) boosting, the reader can find in Appendix C a concrete algorithm incorporating a recursive formula to compute the weights of weak learners. This paves the way of efficient implementations of both accelerated proximal and gradient boosting, as done in the Python package `optboosing`[1].

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. [2016, 2018]:

Regression : $n = 800, d = 100, Y = -\sin(2X^{(1)}) + {X^{(2)}}^2 + X^{(3)} - \exp(-X^{(4)}) + Z_{0.5}$.

Classification : $n = 1500, d = 50, Y = \mathbf{21}_{X^{(1)}+{X^{(4)}}^3+X^{(9)}+\sin(X^{(12)}X^{(18)})+Z_{0.1}>0.38} - 1$, where $\mathbf{1}$ is the indicator function.

The first model covers an additive regression problem, while the second covers a binary classification task with covariate interactions. In both cases, we consider an input random variable $X \in \mathbb{R}^d$, the covariate of which, denoted $(X^{(j)})_{1\leq j\leq d}$, are normally distributed with zero mean and covariance matrix $\Sigma = \left(2^{-|i-j|}\right)_{1\leq i,j\leq d}$. Moreover, in these synthetic models of regression and classification, an additive and independent noise is embodied by the random variable Z_{σ^2}, following a normal distribution with zero mean and variance σ^2.

Four different losses are considered (see Table 1 for a brief description): least squares and least absolute deviations for regression; exponential (with $\beta = 1$) and hinge for classification. Computations for the corresponding (sub)gradients and proximal operators are detailed in Appendix B. On that occasion, it can be remarked that the direction of descent $\mathrm{Prox}_n^{\lambda} C(f_t)$ of proximal boosting applied with the least squares loss is the same as that of gradient boosting, $\widetilde{\nabla}_{\mathrm{n}} C(f_t)$, up to a constant factor (see Appendix 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 1). Nevertheless, since the numerical behaviors are respectively very close to the least absolute deviations and 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 Fig. 2)

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

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	-	$\log_2(1 + \exp(-y y'))$	classification
hinge	-	$\max(0, 1 - y y')$	classification

Table 1: Loss functions.

and 15 (in red in Fig. 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] are included as references. At last, accelerated proximal boosting is left out since convergence is guaranteed neither theoretically, nor numerically (see the next section).

Let us analyze the top panels of Fig. 2: for differentiable losses (least squares and exponential), proximal and gradient descents behave exactly the same (curves with and without the symbol **P** 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). These observations are in line with convergence results of gradient boosting [Grubb and Bagnell, 2011].

Still for differentiable losses, the use of Grubb and Bagnell's residual (the dotted lines in Fig. 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 Fig. 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 Grubb and Bagnell's residual helps gradient boosting to converge. Yet, we remark that residual proximal boosting behaves similarly to proximal boosting (curves are mixed up).

Keeping in mind that proximal boosting is distinguished from gradient boosting for non-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, Fig. 3 depicts the trend of training (top) and test (bottom) losses of proximal boosting for $\lambda \in \{10^{-2}, 10^{-1}, \ldots, 10^{2}\}$ (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).

Fig. 3 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 boosting prevails over gradient boosting from an optimization perspective because it converges faster for sufficiently large λ. Regarding the training loss, the advantage of Grubb and Bagnell's residual is not clear since proximal boosting offers similar convergence rates for large values of λ.

However, the use of Nesterov's acceleration is obvious: it speeds up the decrease of the training and test losses, and thus it makes it possible to build accurate models with very few weak learners.

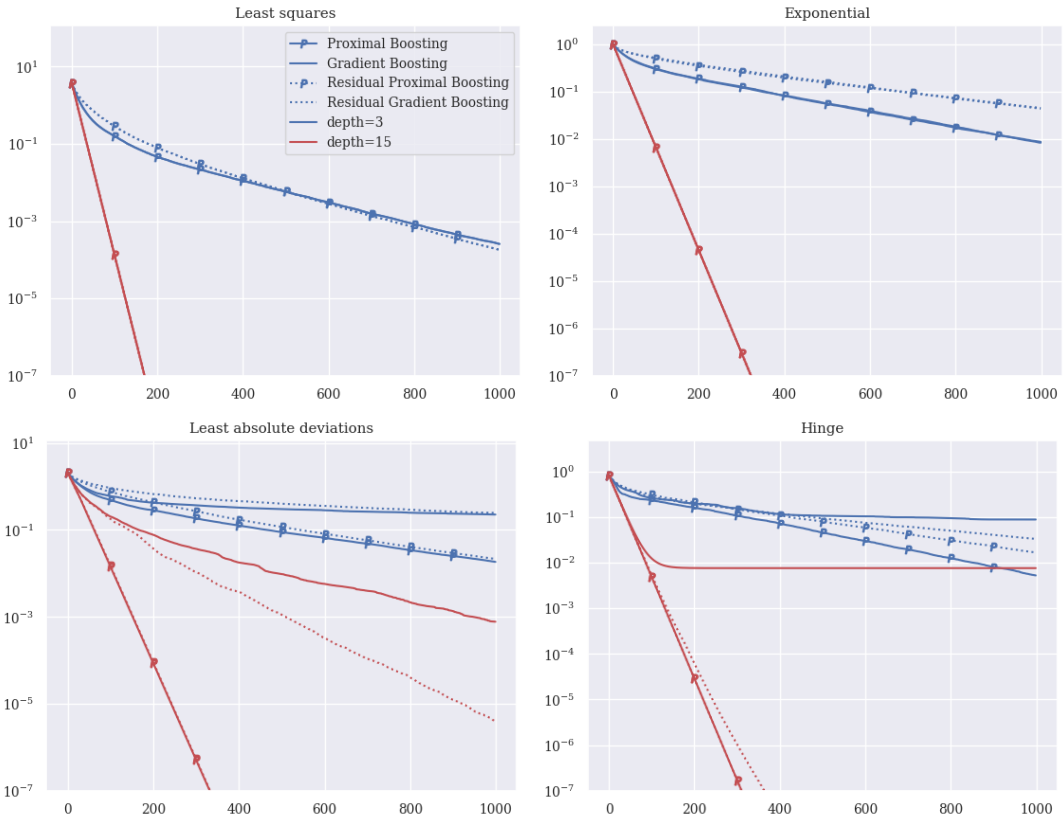

Figure 2: Training losses for two values of maximal depth (3 in blue, 15 in red) vs number of iterations on the horizontal axis.

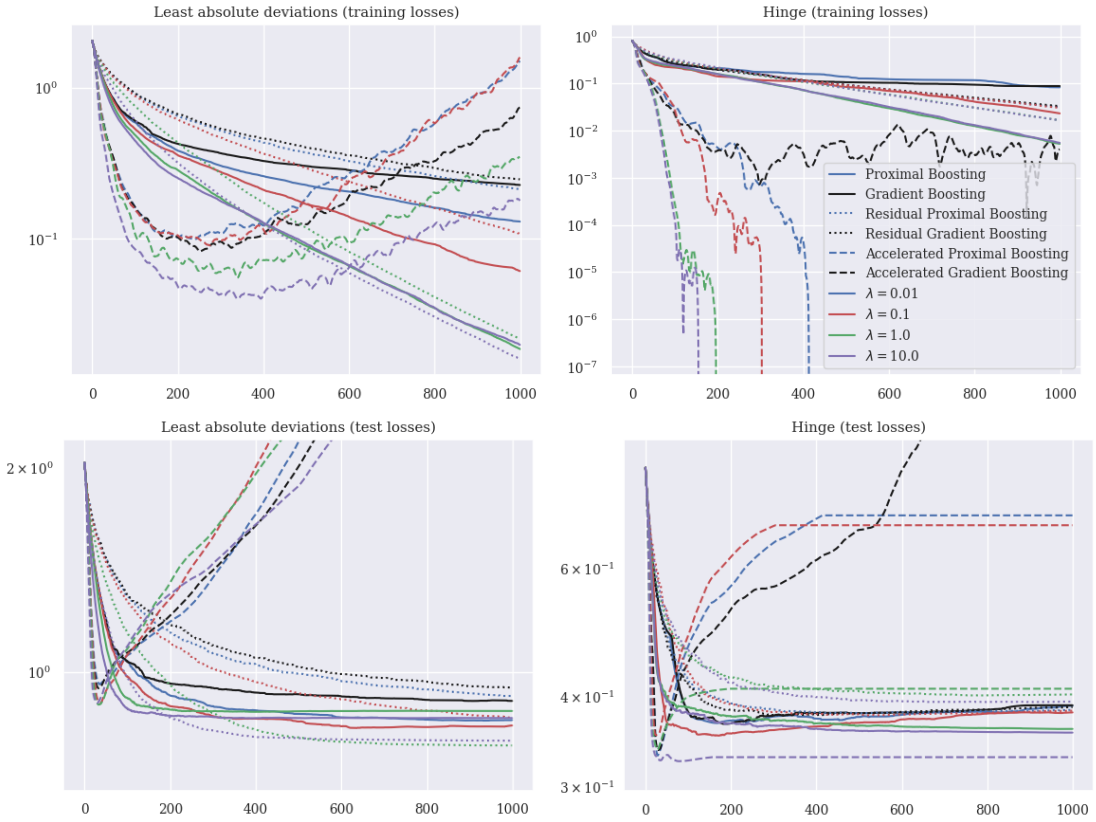

Figure 3: Training (top) and test (bottom) losses of proximal boosting algorithms for several values of the proximal step λ vs number of iterations on the horizontal axis.

Nevertheless, it suffers from instabilities leading to divergence. As observed, this is not a real flaw from the statistical perspective since it always occurs in the overfitting regime.

Comparing test losses, accelerated proximal boosting seems to rarely produce very accurate models (but still better than accelerated gradient boosting). We guess that acceleration makes accelerated boosting very sensitive and dependent on a fine tuning procedure. Yet, proximal and residual proximal boosting often provide the more accurate models and are 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.

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.

Comparison is based on nine datasets (available on the UCI Machine Learning repository), the characteristics of which are described in Table 2. 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 L2 penalty for XGBoost and 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 2: 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 1 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 Fig. 4. $\underline{\Delta \text{ Test loss}}$ 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 that of random forests. Moreover, even though residual boosting was not designed for differentiable losses, it provides the most accurate models for 3 datasets over 6. At last, accelerated versions of boosting methods do not produce more accurate models than vanilla boosting but with about 90% less weak learners (which is a great advantage, already observed in Biau et al. [2018]). A possible explanation concerning the lower perfomance of accelerated boosting is that convergence is so fast, that tuning parameters becomes very tricky.

Looking now at least absolute deviations and pinball losses, we observe that proximal boosting 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.

5.2.2 Classification problems

Losses and misclassification rates computed on the test datasets are depicted respectively in Fig. 5 and in Fig. 6 for the exponential (top) and the hinge (bottom) losses. Besides $\underline{\Delta \text{ Test loss/error}}$, referring to the increment of the loss or misclassification rate from that of gradient boosting, $\underline{\text{Hinge-Exponential}}$ in Fig. 6 represents the increment of the misclassification rate of hinge loss-based boosting from that obtained with the exponential loss.

Regarding both indicators (loss in Fig. 5 and error in Fig. 6), four methods share the winners' podium: proximal boosting (blue), residual proximal boosting (green), residual gradient boosting (brown) 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 (orange) gives sometimes better loss and accuracy than gradient and accelerated gradient boosting (purple). This also comes with less weak learners (even though the comparison is made arduous since maximal depth and learning rate are cross-validated). Both observations confirm the interest of proximal-based boosting for non-differentiable losses.

Very surprisingly, the method performing the best with the hinge loss is XGBoost, while it was not originally designed for non-differentiable losses. It may be explained by advanced optimization tricks included in the last version of the toolbox, offering now the hinge loss as a possible boosting loss. The bottom panel of Fig. 6 shows that, overall using a hinge loss instead of an exponential loss is rarely a big advantage, except to obtain a marginal gain in accuracy with XGBoost on the last two datasets.

6 Conclusion

This paper has introduced a novel boosting algorithm, nicknamed proximal boosting, along with two variants (residual and accelerated proximal boosting), which have appeal for non-differentiable loss functions ℓ. The main idea is to use a proximal-based direction of optimization, which can be coupled with a mechanism of error compensation or Nesterov's acceleration, as already introduced to boosting respectively by Grubb and Bagnell [2011] and Biau et al. [2018]. A theoretical study demonstrates convergence of proximal and residual proximal boosting from an optimization point of view (under different hypotheses on the loss function). As for accelerated gradient boosting, accelerated proximal boosting is observed to diverge but always in the overfitting regime, which is not harmful from a statistical perspective.

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

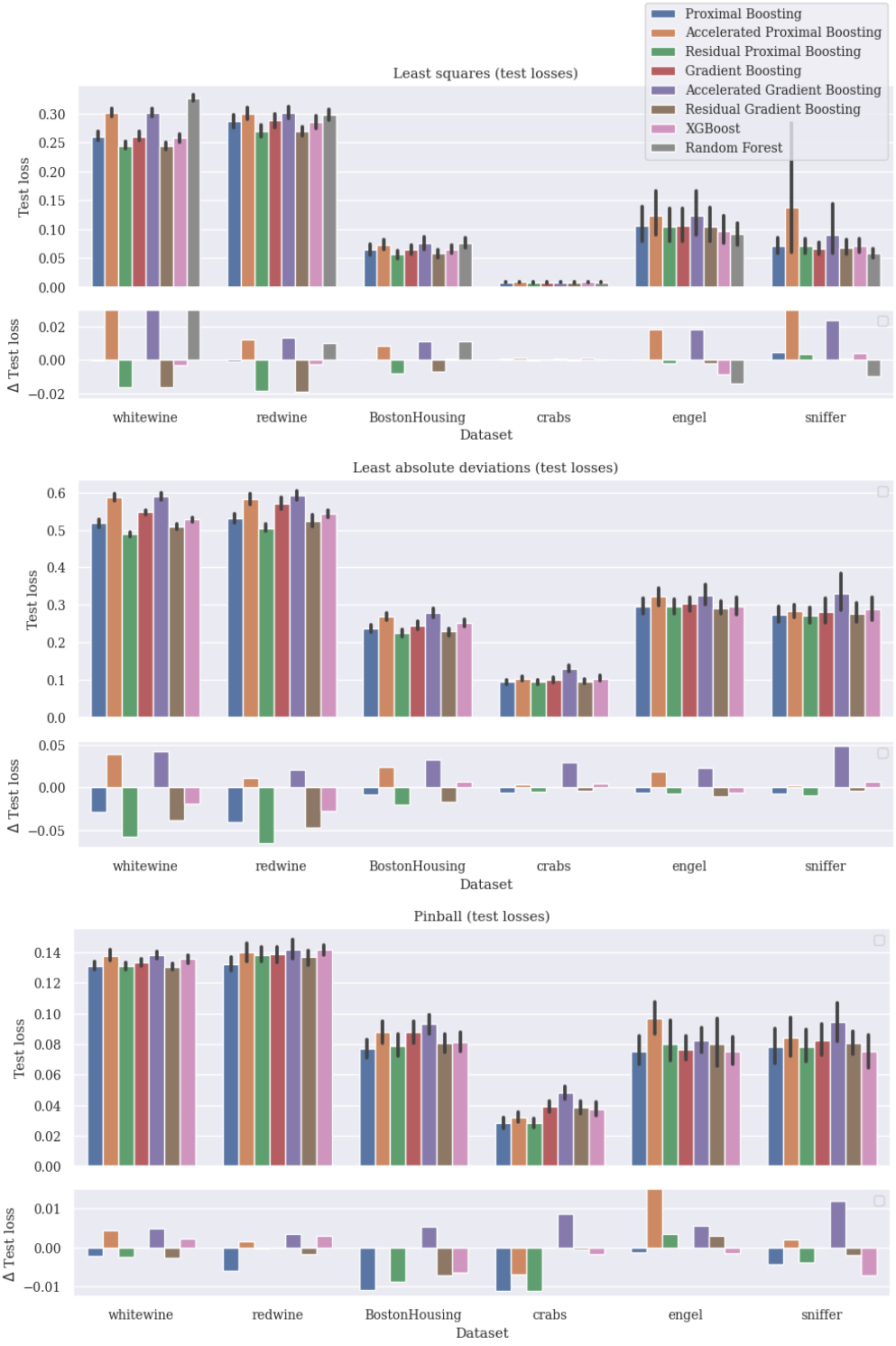

Figure 4: Losses on test datasets for the least squares (top), least absolute deviations (middle) and pinball (bottom) losses. $\underline{\Delta \text{ Test loss}}$ refers to the increment of the loss from that of gradient boosting. The methods proposed in this article are in blue, orange and green.

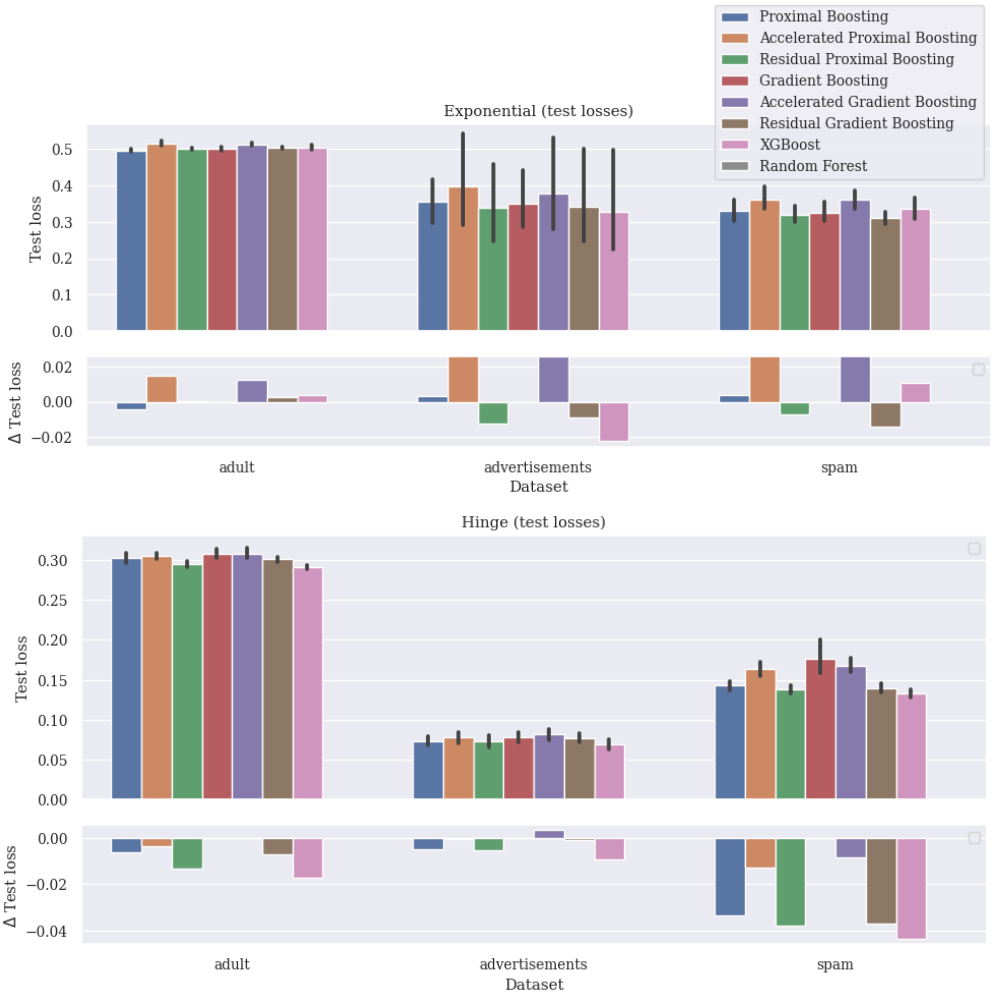

Figure 5: Losses on test datasets for the exponential (top) and hinge (bottom) losses. $\underline{\Delta \text{ Test loss}}$ refers to the increment of the loss from that of gradient boosting. The methods proposed in this article are in blue, orange and green.

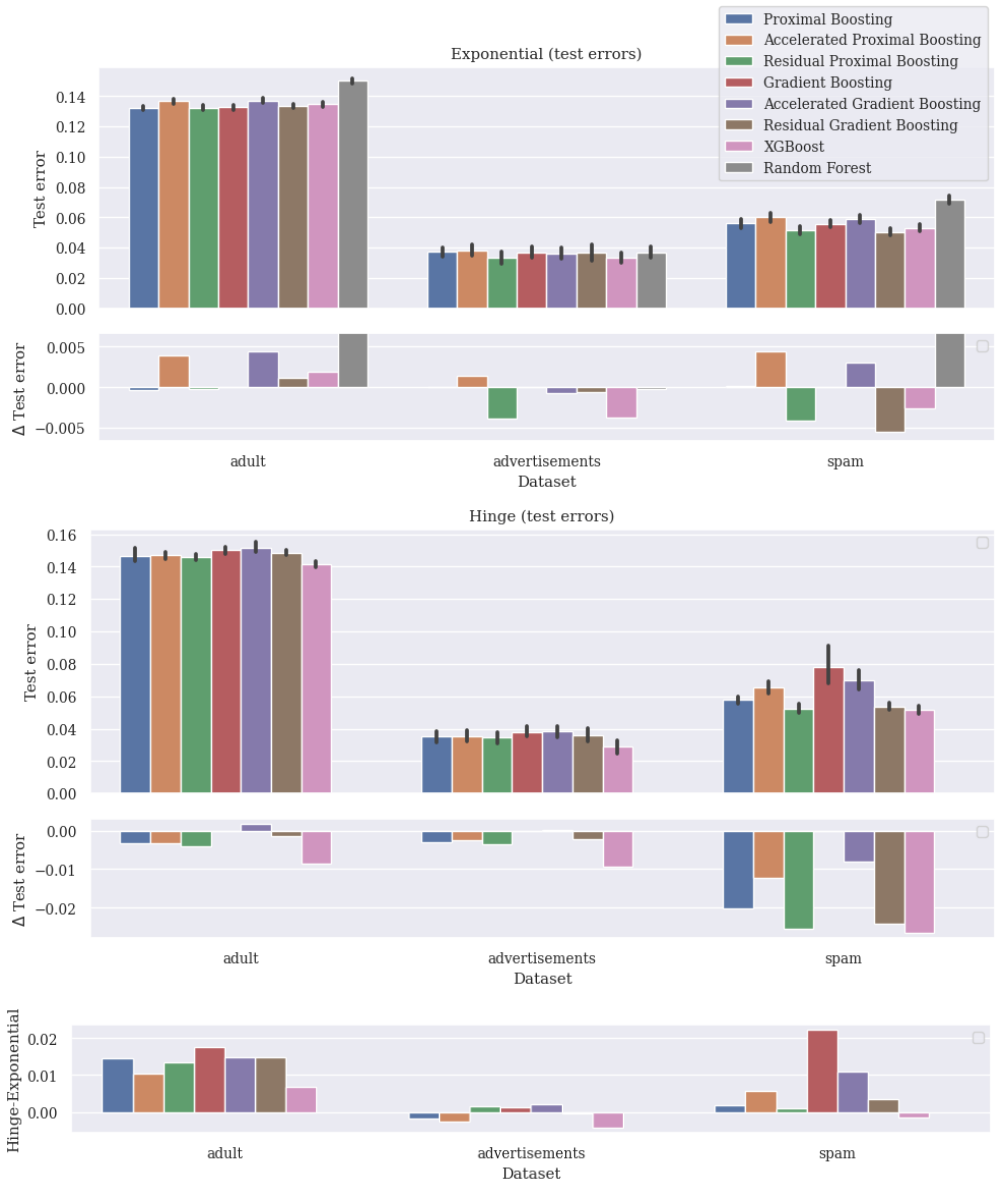

Figure 6: Misclassification rates on test datasets for the exponential (top) and hinge (bottom) losses. Δ Test error and Hinge-Exponential refer to the increment of the misclassification rate respectively from that of gradient boosting and from that obtained with the exponential loss. The methods proposed in this article are in blue, orange and green.

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 situations, proximal or residual proximal boosting often achieve the best test loss for regression and are competitive with residual gradient boosting and XGBoost for classification. On the other hand, similarly to the algorithm introduced in [Biau et al., 2018], accelerated proximal boosting seems quite tricky to tune in order to obtain good generalization results but this is the price to pay for small-sized boosting models.

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

Acknowledgement

The authors are thankful to Gérard Biau and Jalal Fadili for enlightening discussions. They are also indebted to the Associate Editor and the reviewers for suggesting efforts on the theoretical and numerical sides of the paper.

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

A 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 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.

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

$$\forall x, x' \in \mathbb{R}^n, \qquad F(x') \leq F(x) + \langle \nabla F(x), x' - x\rangle_{\ell_2} + \frac{L}{2}\left\|x' - x\right\|_{\ell_2}^2.$$

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.

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

$$\forall x, x' \in \mathbb{R}^n, \forall \eta \in \partial F(x), \qquad F(x') \geq F(x) + \langle \eta, x' - x \rangle_{\ell_2} + \frac{\kappa}{2} \|x' - x\|_{\ell_2}^2 .$$

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

$$\forall x \in \mathbb{R}^n, \forall \eta \in \partial F(x), \qquad \|\eta\|_{\ell_2} \leq G.$$

In any case, it is assumed that:

(E) there exists $\zeta \in (0, 1]$ such that for all $g \in \mathbb{R}^n$, $\|g - P(g)\|_{\ell_2}^2 \leq (1 - \zeta^2) \|g\|_{\ell_2}^2$.

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 directions 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,*

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

Proof. First of all, let us remark that:

1. Assumption (SM) implies L-Lipschitz continuity of ∇F [Nesterov, 2004, Theorem 2.1.5]:

$$\forall x, x' \in \mathbb{R}^n, \qquad \|\nabla F(x) - \nabla F(x')\|_{\ell_2} \leq L \|x - x'\|_{\ell_2} ; \tag{6}$$

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

$$\forall x \in \mathbb{R}^n, \qquad 2\kappa \left(F(x) - F(x^\star)\right) \leq \|\nabla F(x)\|_{\ell_2}^2 . \tag{7}$$

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

$$\begin{aligned} F(x_{t+1}) &\leq F(x_t) + \langle \nabla F(x_t), -\lambda_t P(g_t) \rangle + \frac{L\lambda_t^2}{2} \|P(g_t)\|_{\ell_2}^2 \\ &= F(x_t) - \lambda_t \langle g_t, P(g_t) \rangle - \lambda_t \langle \nabla F(x_t) - g_t, P(g_t) \rangle + \frac{L\lambda_t^2}{2} \|P(g_t)\|_{\ell_2}^2 . \end{aligned} \tag{8}$$

Now, from Assumption (E):

$$\begin{aligned}
-\lambda_t \langle g_t, P(g_t)\rangle &= \frac{\lambda_t}{2}\left(\|g_t - P(g_t)\|_{\ell_2}^2 - \|g_t\|_{\ell_2}^2 - \|P(g_t)\|_{\ell_2}^2\right) \\
&\leq \frac{\lambda_t}{2}\left[(1-\zeta^2)\|g_t\|_{\ell_2}^2 - \|g_t\|_{\ell_2}^2 - \|P(g_t)\|_{\ell_2}^2\right] \\
&= -\frac{\lambda_t \zeta^2}{2}\|g_t\|_{\ell_2}^2 - \frac{\lambda_t}{2}\|P(g_t)\|_{\ell_2}^2 . \qquad (9)
\end{aligned}$$

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

$$\begin{aligned}
\|\nabla F(x_t) - g_t\|_{\ell_2} &= \left\|\nabla F(x_t) - \nabla F\left(\text{prox}_{\lambda_t F}(x_t)\right)\right\|_{\ell_2} \\
&\leq L\left\|x_t - \text{prox}_{\lambda_t F}(x_t)\right\|_{\ell_2} && \text{(Equation (6))} \\
&\leq \lambda_t L \|g_t\|_{\ell_2} && \text{(definition of } g_t\text{).} \qquad (10)
\end{aligned}$$

So,

$$\begin{aligned}
-\lambda_t\langle \nabla F(x_t) - g_t, P(g_t)\rangle &\leq \lambda_t \|\nabla F(x_t) - g_t\|_{\ell_2}\|P(g_t)\|_{\ell_2} && \text{(Cauchy-Schwarz)} \\
&\leq \lambda_t^2 L \|g_t\|_{\ell_2}\|P(g_t)\|_{\ell_2} && \text{(Equation (10))} \\
&\leq 2\lambda_t^2 L \|g_t\|_{\ell_2}^2 , \qquad (11)
\end{aligned}$$

since $\|P(g_t)\|_{\ell_2} \leq 2\|g_t\|_{\ell_2}$, by Assumption (E).

Combining Equations (8), (9) and (11):

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

Now, choosing $\lambda_t = \frac{\zeta^2}{8L}$, one has $\lambda_t\left(\frac{\zeta^2}{2} - 2\lambda_t L\right) = \frac{\zeta^4}{32L}$ and $-\frac{\lambda_t}{2}(1 - L\lambda_t)\|P(g_t)\|_{\ell_2}^2 \leq 0$, leading to:

$$F(x_{t+1}) \leq F(x_t) - \frac{\zeta^4}{32L}\|g_t\|_{\ell_2}^2 . \qquad (12)$$

Let us remark that, by Equation (7):

$$\begin{aligned}
2\kappa\left(F(x_t) - F(x^\star)\right) &\leq \|\nabla F(x_t)\|_{\ell_2}^2 \\
&\leq \left(\|\nabla F(x_t) - g_t\|_{\ell_2} + \|g_t\|_{\ell_2}\right)^2 \\
&\leq (1 + \lambda_t L)^2\|g_t\|_{\ell_2}^2 && \text{(Equation (10))} \\
&\leq \left(1 + \frac{\zeta^2}{8}\right)^2\|g_t\|_{\ell_2}^2 && \left(\lambda_t = \frac{\zeta^2}{8L}\right),
\end{aligned}$$

that is,

$$\|g_t\|_{\ell_2}^2 \geq \frac{128\kappa}{(8+\zeta^2)^2}\left(F(x_t) - F(x^\star)\right).$$

So, from Equation (12),

$$\begin{aligned}
F(x_{t+1}) - F(x^\star) &\leq F(x_t) - F(x^\star) - \frac{\zeta^4}{32L} \|g_t\|_{\ell_2}^2 \\
&\leq \left(1 - \frac{\zeta^4}{32L} \frac{128\kappa}{(8+\zeta^2)^2}\right) (F(x_t) - F(x^\star)) \\
&= \left(1 - \frac{4\zeta^4}{(8+\zeta^2)^2} \frac{\kappa}{L}\right) (F(x_t) - F(x^\star)) \\
&\leq \left(1 - \frac{\zeta^4}{21} \frac{\kappa}{L}\right) (F(x_t) - F(x^\star)) \qquad \left(\forall x \in [0,1], \quad \frac{4x^2}{(8+x)^2} \geq \frac{x^2}{21}\right) \\
&\leq \left(1 - \frac{\zeta^4 \kappa}{21L}\right)^{t+1} (F(x_0) - F(x^\star)) \qquad \text{(by induction).}
\end{aligned}$$

□

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,*

$$\begin{aligned}
\min_{1 \leq t \leq T} F(x_t) - F(x^\star) \leq \frac{1}{2T} \left(\frac{1}{\lambda_0} - \kappa\right) \|x_0 - x^\star\|_{\ell_2}^2 + \frac{1}{2T} \sum_{t=1}^{T-1} \left(\frac{1}{\lambda_t} - \frac{1}{\lambda_{t-1}} - \kappa\right) \|x_t - x^\star\|_{\ell_2}^2 \\
+ \frac{1}{T} \sum_{t=0}^{T-1} \lambda_t \left(\frac{1}{2} \|P(g_t + \Delta_t)\|_{\ell_2}^2 - \left(1 + \frac{\kappa\lambda_t}{2}\right) \|g_t\|_{\ell_2}^2 + \kappa \langle g_t, x_t - x^\star \rangle_{\ell_2} \right. \\
\left. + \langle \Delta_{t+1}, P(g_t + \Delta_t) \rangle_{\ell_2} + G \|g_t - P(g_t + \Delta_t)\|_{\ell_2}\right) + \frac{\lambda_{T-1}}{2T} \|\Delta_T\|_{\ell_2}^2 .
\end{aligned}$$

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

$$\begin{aligned}
F(x^\star) &\geq F(y_{t+1}) + \langle g_t, x^\star - y_{t+1} \rangle_{\ell_2} + \frac{\kappa}{2} \|y_{t+1} - x^\star\|_{\ell_2}^2 \\
&= F(y_{t+1}) + \langle g_t, x^\star - (x_t - \lambda_t g_t) \rangle_{\ell_2} + \frac{\kappa}{2} \|(x_t - \lambda_t g_t) - x^\star\|_{\ell_2}^2 \\
&= F(y_{t+1}) + \langle g_t, x^\star - x_t \rangle_{\ell_2} + \lambda_t \|g_t\|_{\ell_2}^2 + \frac{\kappa}{2} \|x_t - x^\star\|_{\ell_2}^2 + \frac{\kappa\lambda_t^2}{2} \|g_t\|_{\ell_2}^2 - \kappa\lambda_t \langle g_t, x_t - x^\star \rangle_{\ell_2} \\
&= F(y_{t+1}) + \langle P(g_t + \Delta_t), x^\star - x_t \rangle_{\ell_2} + \left(\lambda_t + \frac{\kappa\lambda_t^2}{2}\right) \|g_t\|_{\ell_2}^2 + \frac{\kappa}{2} \|x_t - x^\star\|_{\ell_2}^2 \\
&\quad - \kappa\lambda_t \langle g_t, x_t - x^\star \rangle_{\ell_2} + \langle g_t - P(g_t + \Delta_t), x^\star - x_t \rangle_{\ell_2} . \qquad (13)
\end{aligned}$$

Now, let us analyze the potential $\|x_{t+1} - x^\star\|_{\ell_2}^2$:

$$\begin{aligned}
\|x_{t+1} - x^\star\|_{\ell_2}^2 &= \|x_t - \lambda_t P(g_t + \Delta_t) - x^\star\|_{\ell_2}^2 \\
&= \|x_t - x^\star\|_{\ell_2}^2 + \lambda_t^2 \|P(g_t + \Delta_t)\|_{\ell_2}^2 - 2\lambda_t \langle P(g_t + \Delta_t), x_t - x^\star \rangle_{\ell_2} .
\end{aligned}$$

Thus,

$$\langle P(g_t + \Delta_t), x_t - x^\star \rangle_{\ell_2} = \frac{1}{2\lambda_t} \left(\|x_t - x^\star\|_{\ell_2}^2 - \|x_{t+1} - x^\star\|_{\ell_2}^2\right) + \frac{\lambda_t}{2} \|P(g_t + \Delta_t)\|_{\ell_2}^2 . \qquad (14)$$

Combining Equation (13) and Equation (14), we obtain:

$$\begin{aligned}
F(y_{t+1}) - F(x^\star) &\leq \langle P(g_t + \Delta_t), x_t - x^\star \rangle_{\ell_2} - \left(\lambda_t + \frac{\kappa\lambda_t^2}{2}\right) \|g_t\|_{\ell_2}^2 - \frac{\kappa}{2} \|x_t - x^\star\|_{\ell_2}^2 \\
&\quad + \kappa\lambda_t \langle g_t, x_t - x^\star \rangle_{\ell_2} + \langle g_t - P(g_t + \Delta_t), x_t - x^\star \rangle_{\ell_2} \\
&\leq \frac{1}{2\lambda_t} \left(\|x_t - x^\star\|_{\ell_2}^2 - \|x_{t+1} - x^\star\|_{\ell_2}^2 \right) - \frac{\kappa}{2} \|x_t - x^\star\|_{\ell_2}^2 \\
&\quad + \frac{\lambda_t}{2} \|P(g_t + \Delta_t)\|_{\ell_2}^2 - \left(\lambda_t + \frac{\kappa\lambda_t^2}{2}\right) \|g_t\|_{\ell_2}^2 \\
&\quad + \kappa\lambda_t \langle g_t, x_t - x^\star \rangle_{\ell_2} + \langle g_t - P(g_t + \Delta_t), x_t - x^\star \rangle_{\ell_2} . \qquad (15)
\end{aligned}$$

Now, remark that:

$$\begin{aligned}
&\sum_{t=0}^{T-1} \left(\left(\frac{1}{\lambda_t} - \kappa \right) \|x_t - x^\star\|_{\ell_2}^2 - \frac{1}{\lambda_t} \|x_{t+1} - x^\star\|_{\ell_2}^2 \right) \\
&= \left(\frac{1}{\lambda_0} - \kappa \right) \|x_0 - x^\star\|_{\ell_2}^2 + \sum_{t=1}^{T-1} \left(\frac{1}{\lambda_t} - \kappa \right) \|x_t - x^\star\|_{\ell_2}^2 - \sum_{t=0}^{T-2} \frac{1}{\lambda_t} \|x_{t+1} - x^\star\|_{\ell_2}^2 \\
&- \frac{1}{\lambda_{T-1}} \|x_T - x^\star\|_{\ell_2}^2 \\
&= \left(\frac{1}{\lambda_0} - \kappa \right) \|x_0 - x^\star\|_{\ell_2}^2 + \sum_{t=1}^{T-1} \left(\frac{1}{\lambda_t} - \frac{1}{\lambda_{t-1}} - \kappa \right) \|x_t - x^\star\|_{\ell_2}^2 - \frac{1}{\lambda_{T-1}} \|x_T - x^\star\|_{\ell_2}^2 , \qquad (16)
\end{aligned}$$

and

$$\begin{aligned}
&\sum_{t=0}^{T-1} \langle g_t - P(g_t + \Delta_t), x_t - x^\star \rangle_{\ell_2} \\
&= \sum_{t=0}^{T-1} \langle g_t + \Delta_t - P(g_t + \Delta_t), x_{t+1} + \lambda_t P(g_t + \Delta_t) - x^\star \rangle_{\ell_2} - \sum_{t=0}^{T-1} \langle \Delta_t, x_t - x^\star \rangle_{\ell_2} \\
&= \sum_{t=0}^{T-1} \langle \Delta_{t+1}, x_{t+1} - x^\star \rangle_{\ell_2} - \sum_{t=0}^{T-1} \langle \Delta_t, x_t - x^\star \rangle_{\ell_2} + \sum_{t=0}^{T-1} \lambda_t \langle \Delta_{t+1}, P(g_t + \Delta_t) \rangle_{\ell_2} \\
&= \langle \Delta_T, x_T - x^\star \rangle_{\ell_2} - \langle \Delta_0, x_0 - x^\star \rangle_{\ell_2} + \sum_{t=0}^{T-1} \lambda_t \langle \Delta_{t+1}, P(g_t + \Delta_t) \rangle_{\ell_2} \\
&= \langle \Delta_T, x_T - x^\star \rangle_{\ell_2} + \sum_{t=0}^{T-1} \lambda_t \langle \Delta_{t+1}, P(g_t + \Delta_t) \rangle_{\ell_2} , \qquad (17)
\end{aligned}$$

since $\Delta_0 = 0$.

Then, by summation of Equation (15) and using Equation (16),

$$\begin{aligned}
\sum_{t=0}^{T-1}(F(y_{t+1})-F(x^\star)) &\leq \frac{1}{2}\sum_{t=0}^{T-1}\left(\left(\frac{1}{\lambda_t}-\kappa\right)\|x_t-x^\star\|_{\ell_2}^2-\frac{1}{\lambda_t}\|x_{t+1}-x^\star\|_{\ell_2}^2\right)\\
&+\sum_{t=0}^{T-1}\lambda_t\left(\frac{1}{2}\|P(g_t+\Delta_t)\|_{\ell_2}^2-\left(1+\frac{\kappa\lambda_t}{2}\right)\|g_t\|_{\ell_2}^2+\kappa\langle g_t,x_t-x^\star\rangle_{\ell_2}\right)\\
&+\sum_{t=0}^{T-1}\langle g_t-P(g_t+\Delta_t),x_t-x^\star\rangle_{\ell_2}\\
&=\left(\frac{1}{2\lambda_0}-\frac{\kappa}{2}\right)\|x_0-x^\star\|_{\ell_2}^2+\frac{1}{2}\sum_{t=1}^{T-1}\left(\frac{1}{\lambda_t}-\frac{1}{\lambda_{t-1}}-\kappa\right)\|x_t-x^\star\|_{\ell_2}^2\\
&+\sum_{t=0}^{T-1}\lambda_t\left(\frac{1}{2}\|P(g_t+\Delta_t)\|_{\ell_2}^2-\left(1+\frac{\kappa\lambda_t}{2}\right)\|g_t\|_{\ell_2}^2+\kappa\langle g_t,x_t-x^\star\rangle_{\ell_2}\right)\\
&+\sum_{t=0}^{T-1}\langle g_t-P(g_t+\Delta_t),x_t-x^\star\rangle_{\ell_2}-\frac{1}{2\lambda_{T-1}}\|x_T-x^\star\|_{\ell_2}^2.
\end{aligned}$$

Now, using Equation (17),

$$\begin{aligned}
\sum_{t=0}^{T-1}(F(y_{t+1})-F(x^\star)) &\leq \left(\frac{1}{2\lambda_0}-\frac{\kappa}{2}\right)\|x_0-x^\star\|_{\ell_2}^2+\frac{1}{2}\sum_{t=1}^{T-1}\left(\frac{1}{\lambda_t}-\frac{1}{\lambda_{t-1}}-\kappa\right)\|x_t-x^\star\|_{\ell_2}^2\\
&+\sum_{t=0}^{T-1}\lambda_t\left(\frac{1}{2}\|P(g_t+\Delta_t)\|_{\ell_2}^2-\left(1+\frac{\kappa\lambda_t}{2}\right)\|g_t\|_{\ell_2}^2+\kappa\langle g_t,x_t-x^\star\rangle_{\ell_2}\right.\\
&\left.+\langle\Delta_{t+1},P(g_t+\Delta_t)\rangle_{\ell_2}\right)+\langle\Delta_T,x_T-x^\star\rangle_{\ell_2}-\frac{1}{2\lambda_{T-1}}\|x_T-x^\star\|_{\ell_2}^2\\
&\leq \left(\frac{1}{2\lambda_0}-\frac{\kappa}{2}\right)\|x_0-x^\star\|_{\ell_2}^2+\frac{1}{2}\sum_{t=1}^{T-1}\left(\frac{1}{\lambda_t}-\frac{1}{\lambda_{t-1}}-\kappa\right)\|x_t-x^\star\|_{\ell_2}^2\\
&+\sum_{t=0}^{T-1}\lambda_t\left(\frac{1}{2}\|P(g_t+\Delta_t)\|_{\ell_2}^2-\left(1+\frac{\kappa\lambda_t}{2}\right)\|g_t\|_{\ell_2}^2+\kappa\langle g_t,x_t-x^\star\rangle_{\ell_2}\right.\\
&\left.+\langle\Delta_{t+1},P(g_t+\Delta_t)\rangle_{\ell_2}\right)+\frac{\lambda_{T-1}}{2}\|\Delta_T\|_{\ell_2}^2,
\end{aligned}$$

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,

$$\begin{aligned}
\min_{1\leq t\leq T} F(x_t) - F(x^\star) &\leq \frac{1}{T}\sum_{t=0}^{T-1}\left(F(y_{t+1}) - F(x^\star)\right) + \frac{1}{T}\sum_{t=0}^{T-1}\left(F(x_{t+1}) - F(y_{t+1})\right) \\
&\leq \frac{1}{T}\sum_{t=0}^{T-1}\left(F(y_{t+1}) - F(x^\star)\right) + \frac{1}{T}\sum_{t=0}^{T-1} G\left\|(x_t - \lambda_t P(g_t+\Delta_t)) - (x_t - \lambda_t g_t)\right\|_{\ell_2} \\
&\leq \frac{1}{2T}\left(\frac{1}{\lambda_0} - \kappa\right)\|x_0 - x^\star\|_{\ell_2}^2 + \frac{1}{2T}\sum_{t=1}^{T-1}\left(\frac{1}{\lambda_t} - \frac{1}{\lambda_{t-1}} - \kappa\right)\|x_t - x^\star\|_{\ell_2}^2 \\
&+ \frac{1}{T}\sum_{t=0}^{T-1}\lambda_t\left(\frac{1}{2}\|P(g_t+\Delta_t)\|_{\ell_2}^2 - \left(1+\frac{\kappa\lambda_t}{2}\right)\|g_t\|_{\ell_2}^2 + \kappa\left\langle g_t, x_t - x^\star\right\rangle_{\ell_2}\right. \\
&\left. + \left\langle \Delta_{t+1}, P(g_t+\Delta_t)\right\rangle_{\ell_2} + G\left\|g_t - P(g_t+\Delta_t)\right\|_{\ell_2}\right) + \frac{\lambda_{T-1}}{2T}\|\Delta_T\|_{\ell_2}^2 .
\end{aligned}$$

□

Theorem 6. *Let $(x_t)_t$ be a sequence generated by Algorithm 6. Assume that Assumptions (E) and (L) hold. Assume also that there exists $x^\star \in \arg\min_{x\in\mathbb{R}^n} F(x)$ and that $\|x_t\|_{\ell_2} \leq R$ and $\|x^\star\|_{\ell_2} \leq R$ (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:

$$\begin{aligned}
\min_{1\leq t\leq T} F(x_t) - F(x^\star) &\leq \frac{1}{2\lambda_0 T}\|x_0 - x^\star\|_{\ell_2}^2 + \frac{1}{2T}\sum_{t=1}^{T-1}\left(\frac{1}{\lambda_t} - \frac{1}{\lambda_{t-1}}\right)\|x_t - x^\star\|_{\ell_2}^2 \\
&+ \frac{1}{T}\sum_{t=0}^{T-1}\lambda_t\left(\frac{1}{2}\|P(g_t+\Delta_t)\|_{\ell_2}^2 - \|g_t\|_{\ell_2}^2\right. \\
&\left. + \left\langle \Delta_{t+1}, P(g_t+\Delta_t)\right\rangle_{\ell_2} + G\left\|g_t - P(g_t+\Delta_t)\right\|_{\ell_2}\right) + \frac{\lambda_{T-1}}{2T}\|\Delta_T\|_{\ell_2}^2 \\
&\leq \frac{1}{2T}\sum_{t=0}^{T-1}\left(\sqrt{t+1} - \sqrt{t}\right)\|x_t - x^\star\|_{\ell_2}^2 \\
&+ \frac{1}{T}\sum_{t=0}^{T-1}\frac{1}{\sqrt{t+1}}\left(\frac{1}{2}\|P(g_t+\Delta_t)\|_{\ell_2}^2 + \left\langle \Delta_{t+1}, P(g_t+\Delta_t)\right\rangle_{\ell_2}\right. \\
&\left. + G\left\|g_t - P(g_t+\Delta_t)\right\|_{\ell_2}\right) + \frac{1}{2T^{\frac{3}{2}}}\|\Delta_T\|_{\ell_2}^2 \\
&\leq \frac{2R^2}{\sqrt{T}} + \frac{1}{T}\sum_{t=0}^{T-1}\frac{1}{\sqrt{t+1}}\left(\frac{1}{2}\|P(g_t+\Delta_t)\|_{\ell_2}^2 + \left\langle \Delta_{t+1}, P(g_t+\Delta_t)\right\rangle_{\ell_2}\right. \\
&\left. + G\left\|g_t - P(g_t+\Delta_t)\right\|_{\ell_2}\right) + \frac{1}{2T^{\frac{3}{2}}}\|\Delta_T\|_{\ell_2}^2 .
\end{aligned}$$

Now, since $g_t \in \partial F(y_{t+1})$, $\|g_t\|_{\ell_2} \le G$. In addition, since $\Delta_0 = 0$, by Assumption (E),

$$\begin{aligned}
\|\Delta_{T+1}\|_{\ell_2} &\le \sqrt{1-\zeta^2}\,\|g_T + \Delta_T\|_{\ell_2} \\
&\le \sqrt{1-\zeta^2}\,\|g_T\|_{\ell_2} + \sqrt{1-\zeta^2}\,\|\Delta_T\|_{\ell_2} \\
&\le \sum_{t=0}^{T} \sqrt{1-\zeta^2}^{T+1-t}\,\|g_t\|_{\ell_2} \\
&\le G \sum_{t=1}^{T+1} \sqrt{1-\zeta^2}^{t} \\
&\le \frac{\sqrt{1-\zeta^2}}{1-\sqrt{1-\zeta^2}} G \\
&\le \frac{2}{\zeta^2} G,
\end{aligned}$$

since $\frac{1}{1-\sqrt{1-\zeta^2}} \le \frac{2}{\zeta^2}$. Moreover,

$$\begin{aligned}
\|P(g_t + \Delta_t)\|_{\ell_2} &\le \|P(g_t + \Delta_t) - (g_t + \Delta_t)\|_{\ell_2} + \|g_t + \Delta_t\|_{\ell_2} \\
&\le (\sqrt{1-\zeta^2} + 1)\,\|g_t + \Delta_t\|_{\ell_2} \\
&\le (\sqrt{1-\zeta^2} + 1)G + (\sqrt{1-\zeta^2} + 1)\frac{\sqrt{1-\zeta^2}}{1-\sqrt{1-\zeta^2}} G \\
&\le \frac{(1-(1-\zeta^2)) + (\sqrt{1-\zeta^2} + 1)\sqrt{1-\zeta^2}}{1-\sqrt{1-\zeta^2}} G \\
&\le \frac{\zeta^2 + (1-\zeta^2) + \sqrt{1-\zeta^2}}{1-\sqrt{1-\zeta^2}} G \\
&\le \frac{1+\sqrt{1-\zeta^2}}{1-\sqrt{1-\zeta^2}} G \\
&\le \frac{4}{\zeta^2} G.
\end{aligned}$$

At last,

$$\begin{aligned}
\|g_t - P(g_t + \Delta_t)\|_{\ell_2} &\le \|g_t + \Delta_t - P(g_t + \Delta_t)\|_{\ell_2} + \|\Delta_t\|_{\ell_2} \\
&\le \sqrt{1-\zeta^2}\,\|g_t + \Delta_t\|_{\ell_2} + \|\Delta_t\|_{\ell_2} \\
&\le \sqrt{1-\zeta^2}\,\|g_t\|_{\ell_2} + (\sqrt{1-\zeta^2} + 1)\,\|\Delta_t\|_{\ell_2} \\
&\le \sqrt{1-\zeta^2} G + (\sqrt{1-\zeta^2} + 1)\frac{\sqrt{1-\zeta^2}}{1-\sqrt{1-\zeta^2}} G \\
&\le \frac{(\sqrt{1-\zeta^2} - (1-\zeta^2)) + (1-\zeta^2+\sqrt{1-\zeta^2})}{1-\sqrt{1-\zeta^2}} G \\
&\le \frac{2\sqrt{1-\zeta^2}}{1-\sqrt{1-\zeta^2}} G \\
&\le \frac{4}{\zeta^2} G.
\end{aligned}$$

To conclude,

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

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

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

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

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

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

For each loss, we lay out the following information:

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

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

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

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

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

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

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

B.1 Least squares

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

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

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

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

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

B.2 Least absolute deviations

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

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

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

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

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

B.3 Pinball

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

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

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

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

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

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

Line search: no closed-form solution.

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

Proximal operator: no closed-form solution.

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

Line search: no closed-form solution.

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

Proximal operator: no closed-form solution.

B.6 Hinge loss

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

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

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

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

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

C Accelerated proximal boosting in practice

Algorithm 8 describes a practical version of accelerated proximal boosting (Algorithm 4). 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. As described in Algorithm 8, the line search is only aimed at scaling the weak learner g_{t+1} by a constant factor. However, when the class $\mathcal{F}$ is a set of regression trees, g_{t+1} is a piecewise constant function. In this case, it is common to perform a line search sequentially for each leaf of the decision tree [Friedman, 2001] (called a multiple line search). As a consequence, each level of the piecewise constant function g_{t+1} is scaled with its own factor.

Moreover, Algorithm 8 requires a number of iterations T, which acts on two regularization mechanisms. The first one is statistical (T controls the complexity of the subspace in which f_T lies) and the second one is numerical (T controls the precision to which the empirical risk C is minimized). The shrinkage coefficient ν tunes the balance between these two regularization mechanisms.

Algorithm 8 Accelerated proximal 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 $r \leftarrow \frac{1}{\lambda}\left(\operatorname{prox}_{\lambda D}(v_t) - v_t\right)$ (see Appendix B).
7: Compute $g_{t+1} \in \arg\min_{g \in \mathcal{F}} \|g(X_1^n) - r\|_{\ell_2}$.
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: $f_T = \sum_{t=0}^T w_t^{(T)} g_t$.

In addition, 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: $f_T = \sum_{t=0}^T w_t g_t$. Thus, the weights of the final model have to be tracked despite the recursive update of f_{t+1} (Lines 11 in Algorithm 4 and 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 11 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:

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

It follows that:

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

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

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

□

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. [2018]. This paves the way of efficient implementations of both accelerated gradient and proximal boosting, as done in the Python package `optboosing`[2].

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

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

$$\begin{cases} w_0^{(0)} = 1 \\ w_1^{(0)} = \nu\gamma_1 \\ w_1^{(1)} = \nu\gamma_1 \\ w_j^{(t+1)} = (w_j^{(t)} - w_j^{(t-1)})(1 + \alpha_t) + w_j^{(t-1)}, \forall j \in \{1, \dots, t\} \\ w_{t+1}^{(t+1)} = \nu\gamma_{t+1}. \end{cases} \tag{18}$$

Proof. See proof of Property 7. □