Boosting Frank-Wolfe by Chasing Gradients

Cyrille W. Combettes [1] **Sebastian Pokutta** [2 3]

Abstract

The Frank-Wolfe algorithm has become a popular first-order optimization algorithm for it is simple and projection-free, and it has been successfully applied to a variety of real-world problems. Its main drawback however lies in its convergence rate, which can be excessively slow due to naive descent directions. We propose to speed up the Frank-Wolfe algorithm by better aligning the descent direction with that of the negative gradient via a subroutine. This subroutine chases the negative gradient direction in a matching pursuit-style while still preserving the projection-free property. Although the approach is reasonably natural, it produces very significant results. We derive convergence rates $\mathcal{O}(1/t)$ to $\mathcal{O}(e^{-\omega t})$ of our method and we demonstrate its competitive advantage both per iteration and in CPU time over the state-of-the-art in a series of computational experiments.

1. Introduction

Let $(\mathcal{H}, \langle \cdot, \cdot \rangle)$ be a Euclidean space. In this paper, we address the constrained convex optimization problem

$$\min_{x \in \mathcal{C}} f(x) \tag{1}$$

where $f : \mathcal{H} \to \mathbb{R}$ is a smooth convex function and $\mathcal{C} \subset \mathcal{H}$ is a compact convex set. A natural approach to solving Problem (1) is to apply any efficient method that works in the unconstrained setting and add projections back onto $\mathcal{C}$ when the iterates leave the feasible region. However, there are situations where projections can be very expensive while linear minimizations over $\mathcal{C}$ are much cheaper. For example, if $\mathcal{C} = \{X \in \mathbb{R}^{m \times n} \mid \|X\|_{\text{nuc}} \leqslant \tau\}$ is a nuclear norm-ball, a projection onto $\mathcal{C}$ requires computing an SVD, which has complexity $\mathcal{O}(mn \min\{m, n\})$, while a linear minimization over $\mathcal{C}$ requires only computing the pair of top singular vectors, which has complexity $\mathcal{O}(\text{nnz})$ where nnz denotes the number of nonzero entries. Other examples include the flow polytope, the Birkhoff polytope, the matroid polytope, or the set of rotations; see, e.g., Hazan & Kale (2012).

In these situations, the Frank-Wolfe algorithm (FW) (Frank & Wolfe, 1956), a.k.a. conditional gradient algorithm (Levitin & Polyak, 1966), becomes the method of choice, as it is a simple projection-free algorithm relying on a linear minimization oracle over $\mathcal{C}$. At each iteration, it calls the oracle $v_t \leftarrow \arg\min_{v \in \mathcal{C}} \langle \nabla f(x_t), v \rangle$ and moves in the direction of this vertex, ensuring that the new iterate $x_{t+1} \leftarrow x_t + \gamma_t(v_t - x_t)$ is feasible by convex combination, with a step-size $\gamma_t \in [0, 1]$. Hence, FW can be seen as a projection-free variant of projected gradient descent trading the gradient descent direction $-\nabla f(x_t)$ for the vertex direction $v_t - x_t$ minimizing the linear approximation of f at x_t over $\mathcal{C}$. FW has been applied to traffic assignment problems (LeBlanc et al., 1975), low-rank matrix approximation (Shalev-Shwartz et al., 2011), structural SVMs (Lacoste-Julien et al., 2013), video co-localization (Joulin et al., 2014), infinite RBMs (Ping et al., 2016), and, e.g., adversarial learning (Chen et al., 2020).

The main drawback of FW is that the modified descent direction leads to a sublinear convergence rate $\mathcal{O}(1/t)$, which cannot be improved upon in general as an asymptotic lower bound $\Omega(1/t^{1+\delta})$ holds for any $\delta > 0$ (Canon & Cullum, 1968). More recently, Jaggi (2013) provided a simple illustration of the phenomenon: if $f : x \in \mathbb{R}^n \mapsto \|x\|_2^2$ is the squared ℓ_2-norm and $\mathcal{C} = \Delta_n$ is the standard simplex, then the primal gap at iteration $t \in [\![1, n]\!]$ is lower bounded by $1/t - 1/n$; see also Lan (2013) for a lower bound $\Omega(LD^2/t)$ on an equivalent setup, exhibiting an explicit dependence on the smoothness constant L of f and the diameter D of $\mathcal{C}$.

Hence, a vast literature has been devoted to the analysis of higher convergence rates of FW if additional assumptions on the properties of f, the geometry of $\mathcal{C}$, or the location of $\arg\min_{\mathcal{C}} f$ are met. Important contributions include:

(i) $\mathcal{O}(e^{-\omega t})$ if $\mathcal{C}$ is strongly convex and $\inf_{\mathcal{C}} \|\nabla f\| > 0$

[1]School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, USA [2]Institute of Mathematics, Technische Universität Berlin, Berlin, Germany [3]Department for AI in Society, Science, and Technology, Zuse Institute Berlin, Berlin, Germany. Correspondence to: Cyrille W. Combettes <cyrille@gatech.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

(Levitin & Polyak, 1966),

(ii) $\mathcal{O}(e^{-\omega t})$ if f is strongly convex and $\arg\min_{\mathcal{C}} f \subset \operatorname{relint}(\mathcal{C})$ (Guélat & Marcotte, 1986),

(iii) $\mathcal{O}(1/t^2)$ if f is gradient dominated and $\mathcal{C}$ is strongly convex (Garber & Hazan, 2015).

More recently, several variants to FW have been proposed, achieving linear convergence rates without excessively increasing the per-iteration complexity. These include the following:

(i) $\mathcal{O}(e^{-\omega t})$ when f is strongly convex and $\mathcal{C}$ is a polytope (Garber & Hazan, 2016; Lacoste-Julien & Jaggi, 2015; Braun et al., 2019),

(ii) $\mathcal{O}(e^{-\omega t})$ with constants depending on the sparsity of the solution when f is strongly convex and $\mathcal{C}$ is a polytope, of the form $\{x \in \mathbb{R}^n \mid Ax = b, x \geqslant 0\}$ with vertices in $\{0,1\}^n$ (Garber & Meshi, 2016), or of arbitrary form (Bashiri & Zhang, 2017).

Contributions. We propose the *Boosted Frank-Wolfe* algorithm (BoostFW), a new and intuitive method speeding up the Frank-Wolfe algorithm by chasing the negative gradient direction $-\nabla f(x_t)$ via a matching pursuit-style subroutine, and moving in this better aligned direction. BoostFW thereby mimics gradient descent while remaining projection-free. We derive convergence rates $\mathcal{O}(1/t)$ to $\mathcal{O}(e^{-\omega t})$. Although the linear minimization oracle may be called multiple times per iteration, we demonstrate in a series of computational experiments the competitive advantage both per iteration and in CPU time of our method over the state-of-the-art. Furthermore, BoostFW does not require line search to achieve strong empirical performance, and it does not need to maintain the decomposition of the iterates. Naturally, our approach can also be used to boost the performance of any Frank-Wolfe-style algorithm.

Outline. We start with notation and definitions and we present some background material on the Frank-Wolfe algorithm (Section 2). We then move on to the intuition behind the design of the Boosted Frank-Wolfe algorithm and present its convergence analysis (Section 3). We validate the advantage of our approach in a series of computational experiments (Section 4). Finally, a couple of remarks conclude the paper (Section 5). All proofs are available in Appendix D. The Appendix further contains complementary plots (Appendix A), an application of our approach to boost the Decomposition-Invariant Pairwise Conditional Gradient algorithm (DICG) (Garber & Meshi, 2016) (Appendix B), and the convergence analysis of the line search-free Away-Step Frank-Wolfe algorithm (Appendix C). We were later informed that the latter analysis was already derived by Pedregosa et al. (2020) in a more general setting.

2. Preliminaries

We work in a Euclidean space $(\mathcal{H}, \langle\cdot,\cdot\rangle)$ equipped with the induced norm $\|\cdot\|$. Let $\mathcal{C} \subset \mathcal{H}$ be a nonempty compact convex set. If $\mathcal{C}$ is a polytope, let $\mathcal{V}$ be its set of vertices. Else, slightly abusing notation, we refer to any point in $\mathcal{V} \coloneqq \partial\mathcal{C}$ as a *vertex*. We denote by $D \coloneqq \max_{x,y\in\mathcal{C}} \|y-x\|$ the diameter of $\mathcal{C}$.

2.1. Notation and definitions

For any $i, j \in \mathbb{N}$ satisfying $i \leqslant j$, the brackets $[\![i, j]\!]$ denote the set of integers between (and including) i and j. The indicator function for an event A is $\mathbb{1}_A \coloneqq 1$ if A is true else 0. For any $x \in \mathbb{R}^n$ and $i \in [\![1, n]\!]$, $[x]_i$ denotes the i-th entry of x. Given $p \geqslant 1$, the ℓ_p-norm in $\mathbb{R}^n$ is $\|\cdot\|_p : x \in \mathbb{R}^n \mapsto (\sum_{i=1}^n |[x]_i|^p)^{1/p}$ and the closed ℓ_p-ball of radius $\tau > 0$ is $\mathcal{B}_p(\tau) \coloneqq \{x \in \mathbb{R}^n \mid \|x\|_p \leqslant \tau\}$. The standard simplex in $\mathbb{R}^n$ is $\Delta_n \coloneqq \{x \in \mathbb{R}^n \mid 1^\top x = 1, x \geqslant 0\} = \operatorname{conv}(e_1, \dots, e_n)$ where $\{e_1, \dots, e_n\}$ denotes the standard basis, i.e., $e_i = (\mathbb{1}_{\{1=i\}}, \dots, \mathbb{1}_{\{n=i\}})^\top$. The conical hull of a nonempty set $\mathcal{A} \subseteq \mathcal{H}$ is $\operatorname{cone}(\mathcal{A}) \coloneqq \{\sum_{k=1}^K \lambda_k a_k \mid K \in \mathbb{N}\backslash\{0\}, \lambda_1, \dots, \lambda_K \geqslant 0, a_1, \dots, a_K \in \mathcal{A}\}$. The number of its elements is denoted by $|\mathcal{A}|$.

Let $f : \mathcal{H} \to \mathbb{R}$ be a differentiable function. We say that f is:

(i) L-smooth if $L > 0$ and for all $x, y \in \mathcal{H}$,

$$f(y) - f(x) - \langle \nabla f(x), y - x\rangle \leqslant \frac{L}{2}\|y - x\|^2,$$

(ii) S-strongly convex if $S > 0$ and for all $x, y \in \mathcal{H}$,

$$f(y) - f(x) - \langle \nabla f(x), y - x\rangle \geqslant \frac{S}{2}\|y - x\|^2,$$

(iii) μ-gradient dominated if $\mu > 0$, $\arg\min_{\mathcal{H}} f \neq \varnothing$, and for all $x \in \mathcal{H}$,

$$f(x) - \min_{\mathcal{H}} f \leqslant \frac{\|\nabla f(x)\|^2}{2\mu}.$$

Note that although Definition (iii) is defined with respect to the global optimal value $\min_{\mathcal{H}} f$, the bound holds for the primal gap of f on any compact set $\mathcal{C} \subset \mathcal{H}$:

$$f(x) - \min_{\mathcal{C}} f \leqslant f(x) - \min_{\mathcal{H}} f \leqslant \frac{\|\nabla f(x)\|^2}{2\mu}.$$

Definition (iii) is also commonly referred to as the Polyak-Łojasiewicz inequality or PL inequality (Polyak, 1963; Łojasiewicz, 1963). It is a *local* condition, weaker than that of strong convexity (Fact 2.1), but it can still provide linear convergence rates for non-strongly convex functions (Karimi et al., 2016). For example, the least squares

loss $x \in \mathbb{R}^n \mapsto \|Ax - b\|_2^2$ where $A \in \mathbb{R}^{m \times n}$ and $\operatorname{rank}(A) = m < n$ is not strongly convex, however it is gradient dominated (Garber & Hazan, 2015). See also the Kurdyka-Łojasiewicz inequality (Kurdyka, 1998; Łojasiewicz, 1963) for a generalization to nonsmooth optimization (Bolte et al., 2017).

Fact 2.1. *Let $f : \mathcal{H} \to \mathbb{R}$ be S-strongly convex. Then f is S-gradient dominated.*

2.2. The Frank-Wolfe algorithm

The Frank-Wolfe algorithm (FW) (Frank & Wolfe, 1956), a.k.a. conditional gradient algorithm (Levitin & Polyak, 1966), is presented in Algorithm 1. It is a simple first-order projection-free algorithm relying on a linear minimization oracle over $\mathcal{C}$. At each iteration, it minimizes over $\mathcal{C}$ the linear approximation of f at x_t, i.e., $\ell_f(x_t) : z \in \mathcal{C} \mapsto f(x_t) + \langle \nabla f(x_t), z - x_t \rangle$, by calling the oracle (Line 2) and moves in that direction by convex combination (Line 3). Hence, the new iterate x_{t+1} is guaranteed to be feasible by convexity and there is no need to use projections back onto $\mathcal{C}$. In short, FW solves Problem (1) by minimizing a sequence of linear approximations of f over $\mathcal{C}$.

Algorithm 1 Frank-Wolfe (FW)

Input: Start point $x_0 \in \mathcal{C}$, step-size strategy $\gamma_t \in [0, 1]$.
Output: Point $x_T \in \mathcal{C}$.

1: **for** $t = 0$ **to** $T - 1$ **do**
2: $\quad v_t \leftarrow \arg\min_{v \in \mathcal{V}} \langle \nabla f(x_t), v \rangle$ ▷ FW oracle
3: $\quad x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$
4: **end for**

Note that FW has access to the feasible region $\mathcal{C}$ only via the linear minimization oracle, which receives any $c \in \mathcal{H}$ as input and outputs a point $v \in \arg\min_{z \in \mathcal{C}} \langle c, z \rangle = \arg\min_{v \in \mathcal{V}} \langle c, v \rangle$. For example, if $\mathcal{H} = \mathbb{R}^n$ and $\mathcal{C} = \{x \in \mathbb{R}^n \mid \|x\|_1 \leqslant \tau\}$ is an ℓ_1-ball, then $\mathcal{V} = \{\pm \tau e_1, \dots, \pm \tau e_n\}$ so the linear minimization oracle simply picks the coordinate e_i with the largest absolute magnitude $|[c]_i|$ and returns $-\operatorname{sign}([c]_i) \tau e_i$. In this case, FW accesses $\mathcal{C}$ only by reading coordinates. Some other examples are covered in the experiments (Section 4).

The general convergence rate of FW is $\mathcal{O}(LD^2/t)$, where L is the smoothness constant of f and D is the diameter of $\mathcal{C}$ (Levitin & Polyak, 1966; Jaggi, 2013). There are different step-size strategies possible to achieve this rate. The default strategy is $\gamma_t \leftarrow 2/(t+2)$. It is very simple to implement but it does not guarantee progress at each iteration. The next strategy, sometimes referred to as the *short step* strategy and which does make FW a descent algorithm, is $\gamma_t \leftarrow \min\{\langle \nabla f(x_t), x_t - v_t \rangle / (L\|x_t - v_t\|^2), 1\}$. It minimizes the quadratic smoothness upper bound on f. If $\varepsilon_t \coloneqq f(x_t) - \min_{\mathcal{C}} f$ denotes the primal gap, then

$$\varepsilon_{t+1} \leqslant \begin{cases} \varepsilon_t - \dfrac{\langle \nabla f(x_t), x_t - v_t \rangle^2}{2L\|x_t - v_t\|^2} & \text{if } \gamma_t < 1 \\ \varepsilon_t / 2 & \text{if } \gamma_t = 1. \end{cases}$$

As we can already see here, a quadratic improvement in progress is obtained if the direction $v_t - x_t$ in which FW moves is better aligned with that of the negative gradient $-\nabla f(x_t)$. The third step-size strategy is a line search $\gamma_t \leftarrow \arg\min_{\gamma \in [0,1]} f(x_t + \gamma(v_t - x_t))$. It is the most expensive strategy but it does not require (approximate) knowledge of L and it often yields more progress per iteration in practice.

3. Boosting Frank-Wolfe

3.1. Motivation

Suppose that $\mathcal{C}$ is a polytope and that the set of global minimizers $\arg\min_{\mathcal{H}} f$ lies on a lower dimensional face. Then FW can be very slow to converge as it is allowed only to follow vertex directions. As a simple illustration, consider the problem of minimizing $f : x \in \mathbb{R}^2 \mapsto \|x\|_2^2/2$ over the convex hull of $\{(-1, 0)^\top, (1, 0)^\top, (0, 1)^\top\}$, starting from $x_0 = (0, 1)^\top$. The minimizer is $x^* = (0, 0)^\top$. We computed the first iterates of FW and we present their trajectory in Figure 1. We can see that the iterates try to reach x^* by moving *towards* vertices but clearly these directions $v_t - x_t$ are inadequate as they become orthogonal to $x^* - x_t$.

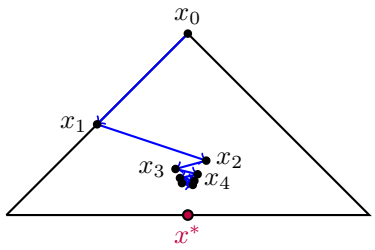

Figure 1. FW yields an inefficient zig-zagging trajectory towards the minimizer.

To remedy this phenomenon, Wolfe (1970) proposed the Away-Step Frank-Wolfe algorithm (AFW), a variant of FW that allows to move *away* from vertices. The issue in Figure 1 is that the iterates are held back by the weight of vertex x_0 in their convex decomposition. Figure 2 shows that AFW is able to remove this weight and thereby to converge much faster to x^*. In fact, Lacoste-Julien & Jaggi (2015) established that AFW with line search converges at a linear rate $\mathcal{O}\big(LD^2 \exp(-(S/(8L))(W/D)^2 t)\big)$ for S-strongly convex functions over polytopes, where W is the *pyramidal width* of the polytope.

However, these descent directions are still not as favorable as those of gradient descent, the pyramidal width is a

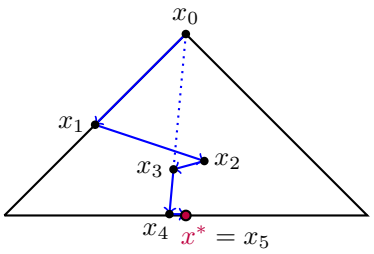

Figure 2. AFW breaks the zig-zagging trajectory by performing away steps. Here, x_4 is obtained using an away step which enables $x_5 = x^*$, speeding up the algorithm considerably.

dimension-dependent quantity, and AFW further requires to maintain the decomposition of the iterates onto $\mathcal{V}$ which can become very expensive both in memory usage and computation time (Garber & Meshi, 2016). Thus, we aim at improving the FW descent direction by directly estimating the gradient descent direction $-\nabla f(x_t)$ using $\mathcal{V}$, in order to maintain the projection-free property. Suppose that $-\nabla f(x_t) \in \operatorname{cone}(\mathcal{V} - x_t)$ and that we are able to compute its conical decomposition, i.e., we have $-\nabla f(x_t) = \sum_{k=0}^{K_t-1} \lambda_k (v_k - x_t)$ where $\lambda_0, \dots, \lambda_{K_t-1} > 0$ and $v_0, \dots, v_{K_t-1} \in \mathcal{V}$. Then by normalizing by $\Lambda_t \coloneqq \sum_{k=0}^{K_t-1} \lambda_k$, we obtain a *feasible* descent direction $g_t \coloneqq (1/\Lambda_t) \sum_{k=0}^{K_t-1} \lambda_k (v_k - x_t)$ in the sense that $[x_t, x_t + g_t] \subseteq \mathcal{C}$. Therefore, building x_{t+1} as a convex combination of x_t and $x_t + g_t$ ensures that $x_{t+1} \in \mathcal{C}$ and the projection-free property holds as in a typical FW step, all the while moving in the direction of the negative gradient $-\nabla f(x_t)$.

3.2. Boosting via gradient pursuit

In practice however, computing the exact conical decomposition of $-\nabla f(x_t)$, even when this is feasible, is not necessary and it may be overkill. Indeed, all we want is to find a descent direction g_t using $\mathcal{V}$ that is *better* aligned with $-\nabla f(x_t)$ and we do not mind if $\| - \nabla f(x_t) - g_t \|$ is arbitrarily large. Thus, we propose to chase the direction of $-\nabla f(x_t)$ by sequentially picking up vertices in a matching pursuit-style (Mallat & Zhang, 1993). The procedure is described in Algorithm 2 (Lines 3-19). In fact, it implicitly addresses the cone constrained quadratic optimization subproblem

$$\min_{d \in \operatorname{cone}(\mathcal{V} - x_t)} \frac{1}{2} \| - \nabla f(x_t) - d \|^2 \tag{2}$$

via the Non-Negative Matching Pursuit algorithm (NNMP) (Locatello et al., 2017), without however the aim of solving it. At each round k, the procedure looks to reduce the residual r_k by subtracting its projection $\lambda_k u_k$ onto the principal component u_k. The comparison $\langle r_k, v_k - x_t \rangle$ vs. $\langle r_k, -d_k / \|d_k\| \rangle$ in Line 9 is less intuitive than the rest of the procedure but it is necessary to ensure convergence; see Locatello et al. (2017). The normalization in Line 21 ensures the feasibility of the new iterate x_{t+1}.

Algorithm 2 Boosted Frank-Wolfe (BoostFW)

Input: Input point $y \in \mathcal{C}$, maximum number of rounds $K \in \mathbb{N} \backslash \{0\}$, alignment improvement tolerance $\delta \in]0, 1[$, step-size strategy $\gamma_t \in [0, 1]$.
Output: Point $x_T \in \mathcal{C}$.

1: $x_0 \leftarrow \arg\min_{v \in \mathcal{V}} \langle \nabla f(y), v \rangle$
2: **for** $t = 0$ **to** $T - 1$ **do**
3: $\quad d_0 \leftarrow 0$
4: $\quad \Lambda_t \leftarrow 0$
5: $\quad$ flag $\leftarrow$ **false**
6: $\quad$ **for** $k = 0$ **to** $K - 1$ **do**
7: $\quad\quad r_k \leftarrow -\nabla f(x_t) - d_k$ ▷ k-th residual
8: $\quad\quad v_k \leftarrow \arg\max_{v \in \mathcal{V}} \langle r_k, v \rangle$ ▷ FW oracle
9: $\quad\quad u_k \leftarrow \arg\max_{u \in \{v_k - x_t, -d_k / \|d_k\|\}} \langle r_k, u \rangle$
10: $\quad\quad \lambda_k \leftarrow \dfrac{\langle r_k, u_k \rangle}{\|u_k\|^2}$
11: $\quad\quad d'_k \leftarrow d_k + \lambda_k u_k$
12: $\quad\quad$ **if** $\operatorname{align}(-\nabla f(x_t), d'_k) - \operatorname{align}(-\nabla f(x_t), d_k) \geqslant \delta$ **then**
13: $\quad\quad\quad d_{k+1} \leftarrow d'_k$
14: $\quad\quad\quad \Lambda_t \leftarrow \begin{cases} \Lambda_t + \lambda_k & \text{if } u_k = v_k - x_t \\ \Lambda_t (1 - \lambda_k / \|d_k\|) & \text{if } u_k = -d_k / \|d_k\| \end{cases}$
15: $\quad\quad$ **else**
16: $\quad\quad\quad$ flag $\leftarrow$ **true**
17: $\quad\quad\quad$ **break** ▷ exit k-loop
18: $\quad\quad$ **end if**
19: $\quad$ **end for**
20: $\quad K_t \leftarrow k$ **if** flag $=$ **true else** K
21: $\quad g_t \leftarrow d_{K_t} / \Lambda_t$ ▷ normalization
22: $\quad x_{t+1} \leftarrow x_t + \gamma_t g_t$
23: **end for**

Since we are only interested in the direction of $-\nabla f(x_t)$, the stopping criterion in the procedure (Line 12) is an alignment condition between $-\nabla f(x_t)$ and the current estimated direction d_k, which serves as descent direction for BoostFW. The function align, defined in (3), measures the alignment between a target direction $d \in \mathcal{H} \backslash \{0\}$ and its estimate $\hat{d} \in \mathcal{H}$. It is invariant by scaling of d or $\hat{d}$, and the higher the value, the better the alignment:

$$\operatorname{align}(d, \hat{d}) \coloneqq \begin{cases} \dfrac{\langle d, \hat{d} \rangle}{\|d\| \|\hat{d}\|} & \text{if } \hat{d} \neq 0 \\ -1 & \text{if } \hat{d} = 0. \end{cases} \tag{3}$$

In order to optimize the trade-off between progress and complexity per iteration, we allow for (very) inexact alignments and we stop the procedure as soon as *sufficient*

progress is not met (Lines 15-17). Furthermore, note that it is not possible to obtain a perfect alignment when $-\nabla f(x_t) \notin \text{cone}(\mathcal{V} - x_t)$, but this is not an issue as we only seek to better align the descent direction. The number of pursuit rounds at iteration t is denoted by K_t (Line 20). In the experiments (Section 4), we typically set $\delta \leftarrow 10^{-3}$ and $K \leftarrow +\infty$; the role of K is only to cap the number of pursuit rounds per iteration when the FW oracle is particularly expensive (see Section 4.3). Note that if $K = 1$ then BoostFW reduces to FW.

In the case of Figures 1-2, BoostFW exactly estimates the direction of $-\nabla f(x_0) = -(x_0 - x^*)$ in only two rounds and converges in 1 iteration. A more general illustration of the procedure is presented in Figure 3. See also Appendix A.2 for an illustration of the improvements in alignment of d_k during the procedure. Lastly, note that BoostFW does not need to maintain the decomposition of the iterates, which is very favorable in practice (Garber & Meshi, 2016).

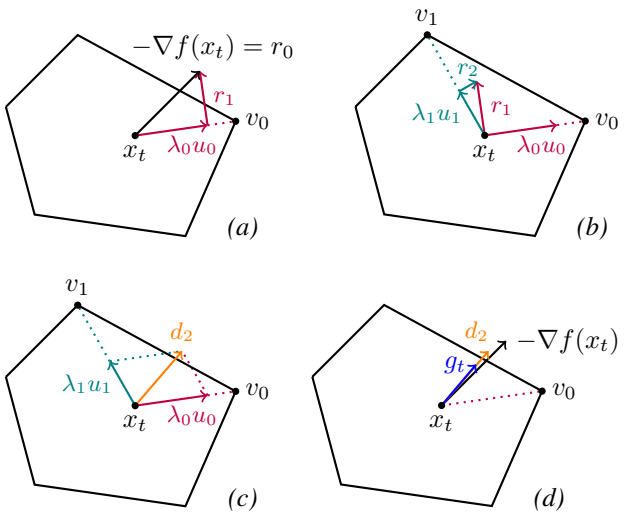

Figure 3. The gradient pursuit procedure builds a descent direction g_t better aligned with the negative gradient direction $-\nabla f(x_t)$, while the FW descent direction is that of $v_0 - x_t$. We have $g_t = d_2/(\lambda_0 + \lambda_1)$ where $d_2 = \lambda_0 u_0 + \lambda_1 u_1$, $u_0 = v_0 - x_t$, and $u_1 = v_1 - x_t$. Furthermore, note that $[x_t, x_t + d_2] \not\subseteq \mathcal{C}$ but $[x_t, x_t + g_t] \subseteq \mathcal{C}$. Moving along the segment $[x_t, x_t + g_t]$ ensures feasibility of the new iterate x_{t+1}.

We present in Proposition 3.1 some properties satisfied by BoostFW (Algorithm 2). Proofs are available in Appendix D.2.

Proposition 3.1. *For all* $t \in [\![0, T-1]\!]$,

(i) d_1 *is defined and* $K_t \geqslant 1$,

(ii) $\lambda_0, \dots, \lambda_{K_t - 1} \geqslant 0$,

(iii) $d_k \in \text{cone}(\mathcal{V} - x_t)$ *for all* $k \in [\![0, K_t]\!]$,

(iv) $x_t + g_t \in \mathcal{C}$ *and* $x_{t+1} \in \mathcal{C}$,

(v) $\text{align}(-\nabla f(x_t), g_t) \geqslant \text{align}(-\nabla f(x_t), v_t - x_t) + (K_t - 1)\delta$ *where* $v_t \in \arg\min_{v \in \mathcal{V}} \langle \nabla f(x_t), v \rangle$ *and* $\text{align}(-\nabla f(x_t), v_t - x_t) \geqslant 0$.

3.3. Convergence analysis

We denote by $\eta_t \coloneqq \text{align}(-\nabla f(x_t), g_t)$. We provide in Theorem 3.2 the general convergence rate of BoostFW. All proofs are available in Appendix D.3. Note that $\eta_t \|\nabla f(x_t)\| / (L\|g_t\|) = \langle -\nabla f(x_t), g_t \rangle / (L\|g_t\|^2)$ corresponds to the short step strategy.

Theorem 3.2 (Universal rate)**.** *Let* $f : \mathcal{H} \to \mathbb{R}$ *be* L*-smooth, convex, and* μ*-gradient dominated, and set* $\gamma_t \leftarrow \min\{\eta_t \|\nabla f(x_t)\| / (L\|g_t\|), 1\}$ *or* $\gamma_t \leftarrow \arg\min_{\gamma \in [0,1]} f(x_t + \gamma g_t)$*. Then for all* $t \in [\![0, T]\!]$,

$$f(x_t) - \min_{\mathcal{C}} f \leqslant \frac{LD^2}{2} \prod_{s=0}^{t-1} \left(1 - \eta_s^2 \frac{\mu}{L}\right)^{\mathbb{1}_{\{\gamma_s < 1\}}} \left(1 - \frac{\|g_s\|}{2\|v_s - x_s\|}\right)^{\mathbb{1}_{\{\gamma_s = 1\}}}$$

where $v_s \in \arg\min_{v \in \mathcal{V}} \langle \nabla f(x_s), v \rangle$ *for all* $s \in [\![0, T-1]\!]$.

Strictly speaking, the rate in Theorem 3.2 is not *explicit* although it still provides a quantitative estimation. Note that $\gamma_t = 1$ is extremely rare in practice, and we observed no more than 1 such iteration in each of the experiments (Section 4). This is a similar phenomenon to that in the Away-Step and Pairwise Frank-Wolfe algorithms (Lacoste-Julien & Jaggi, 2015). Similarly, $K_t > 1$ simply means that it is possible to increase the alignment by δ twice and consecutively, where δ is typically set to a low value. In the experiments, we set $\delta \leftarrow 10^{-3}$ and we observed $K_t > 1$ (or even $K_t > 5$) almost everytime.

For completeness, we disregard these observations and address in Theorem 3.3 the case where the number of iterations with $\gamma_t < 1$ and $K_t > 1$ is not dominant, and we add a minor adjustment to Algorithm 2: if $\gamma_t = 1$ then we choose to do a simple FW step, i.e., to move in the direction of $v_{k=0} - x_t$ instead of the direction of g_t, where $v_{k=0}$ is computed in the first round of the procedure (Line 8). Although this usually provides less progress, we do it for the sole purpose of presenting a *fully explicit* convergence rate; again, there is no need for such tweaks in practice as typically almost every iteration satisfies $\gamma_t < 1$ and $K_t > 1$. Theorem 3.3 states the convergence rate for this scenario, which is very loose as it accommodates for these FW steps.

Theorem 3.3 (Worst-case rate)**.** *Let* $f : \mathcal{H} \to \mathbb{R}$ *be* L*-smooth, convex, and* μ*-gradient dominated, and set* $\gamma_t \leftarrow \min\{\eta_t \|\nabla f(x_t)\| / (L\|g_t\|), 1\}$ *or* $\gamma_t \leftarrow \arg\min_{\gamma \in [0,1]} f(x_t + \gamma g_t)$*. Consider Algorithm 2 with the minor adjustment* $x_{t+1} \leftarrow x_t + \gamma_t'(v_t - x_t)$ *in Line 22*

when $\gamma_t = 1$, *where* $v_t \leftarrow v_{k=0}$ *is computed in Line 8 and* $\gamma'_t \leftarrow \min\{\langle \nabla f(x_t), x_t - v_t\rangle / (L\|x_t - v_t\|^2), 1\}$ *or* $\gamma'_t \leftarrow \arg\min_{\gamma\in[0,1]} f(x_t + \gamma(v_t - x_t))$. *Then for all* $t \in [\![0, T]\!]$,

$$f(x_t) - \min_{\mathcal{C}} f \leqslant \frac{4LD^2}{t+2}.$$

We now provide in Theorem 3.4 the more realistic convergence rate of BoostFW, where $N_t := |\{s \in [\![0, t-1]\!] \mid \gamma_s < 1, K_s > 1\}|$ is *nonnegligeable*, i.e., $N_t \geqslant \omega t^p$ for some $\omega > 0$ and $p \in]0, 1]$. This is the rate observed in practice, where $N_t \approx t - 1$ so $\omega \lesssim 1$ and $p = 1$ (Section 4).

Theorem 3.4 (Practical rate)**.** *Let* $f : \mathcal{H} \to \mathbb{R}$ *be* L*-smooth, convex, and* μ*-gradient dominated, and set* $\gamma_t \leftarrow \min\{\eta_t \|\nabla f(x_t)\| / (L\|g_t\|), 1\}$ *or* $\gamma_t \leftarrow \arg\min_{\gamma\in[0,1]} f(x_t + \gamma g_t)$. *Assume that* $|\{s \in [\![0, t-1]\!] \mid \gamma_s < 1, K_s > 1\}| \geqslant \omega t^p$ *for all* $t \in [\![0, T-1]\!]$, *for some* $\omega > 0$ *and* $p \in]0, 1]$. *Then for all* $t \in [\![0, T]\!]$,

$$f(x_t) - \min_{\mathcal{C}} f \leqslant \frac{LD^2}{2} \exp\left(-\delta^2 \frac{\mu}{L} \omega t^p\right).$$

Remark 3.5. *Note that when* $\gamma_t < 1$ *and* $K_t > 1$, *we have (see proofs in Appendix D.3)*

$$\left(f(x_t) - \min_{\mathcal{C}} f\right) - \left(f(x_{t+1}) - \min_{\mathcal{C}} f\right) \geqslant \delta^2 \frac{\|\nabla f(x_t)\|^2}{2L}$$

so if $N_T := |\{t \in [\![0, T-1]\!] \mid \gamma_t < 1, K_t > 1\}|$, *then*

$$f(x_0) - \min_{\mathcal{C}} f \geqslant \delta^2 \frac{\inf_{\mathcal{C}} \|\nabla f\|^2}{2L} N_T.$$

Thus, if $\inf_{\mathcal{C}} \|\nabla f\| > 0$ *then*

$$N_T \leqslant \frac{2L(f(x_0) - \min_{\mathcal{C}} f)}{\delta^2 \inf_{\mathcal{C}} \|\nabla f\|^2} \leqslant \left(\frac{LD}{\delta \inf_{\mathcal{C}} \|\nabla f\|}\right)^2$$

since $f(x_0) - \min_{\mathcal{C}} f \leqslant LD^2/2$ *(see proofs in Appendix D.3). However, the assumption in Theorem 3.4 can still hold as convergence is usually achieved within* T *iterations where*

$$T = \mathcal{O}\left(\left(\frac{1}{\omega}\left(\frac{LD}{\delta \inf_{\mathcal{C}} \|\nabla f\|}\right)^2\right)^{1/p}\right)$$

for some $\omega > 0$ *and* $p \in]0, 1]$. *In the experiments for example (Section 4), convergence is always achieved within* $\mathcal{O}(10^3)$ *iterations. Furthermore, early stopping to increase the generalization error of a model also prevents* T *from blowing up.*

Lastly, we provide in Corollary 3.6 a bound on the number of FW oracle calls, i.e., the number of linear minimizations over $\mathcal{C}$, performed to achieve ε-convergence. In comparison, FW and AFW respectively require $\mathcal{O}(LD^2/\varepsilon)$ and $\mathcal{O}\big((L/S)(D/W)^2 \ln(1/\varepsilon)\big)$ oracle calls, where f is assumed to be S-strongly convex and $\mathcal{C}$ is assumed to be a polytope with pyramidal width W for AFW (Lacoste-Julien & Jaggi, 2015). It is clear from its design that BoostFW performs more oracle calls per iteration, however it uses them more efficiently and the progress obtained overcomes the cost. This is demonstrated in the experiments (Section 4).

Corollary 3.6. *In order to achieve* ε*-convergence, the number of linear minimizations performed over* $\mathcal{C}$ *is*

$$\begin{cases} \mathcal{O}\left(\dfrac{LD^2 \min\{K, 1/\delta\}}{\varepsilon}\right) & \text{in the worst-case scenario} \\ \mathcal{O}\left(\min\left\{K, \dfrac{1}{\delta}\right\}\left(\dfrac{1}{\omega\delta^2}\dfrac{L}{\mu}\ln\left(\dfrac{1}{\varepsilon}\right)\right)^{1/p}\right) & \\ & \text{in the practical scenario.} \end{cases}$$

Note that the practical scenario assumes that we have set $K \geqslant 2$ *in BoostFW (*$K = 1$ *reduces BoostFW to FW).*

4. Computational experiments

We compared the Boosted Frank-Wolfe algorithm (BoostFW, Algorithm 2) to the Away-Step Frank-Wolfe algorithm (AFW) (Wolfe, 1970), the Decomposition-Invariant Pairwise Conditional Gradient algorithm (DICG) (Garber & Meshi, 2016), and the Blended Conditional Gradients algorithm (BCG) (Braun et al., 2019) in a series of computational experiments. We ran two strategies for AFW, one with the default line search (AFW-ls) and one using the smoothness of f (AFW-L):

$$\gamma_t \leftarrow \begin{cases} \min\left\{\dfrac{\langle \nabla f(x_t), x_t - v_t^{\text{FW}}\rangle}{L\|x_t - v_t^{\text{FW}}\|_2^2}, 1\right\} & \text{if FW step} \\ \min\left\{\dfrac{\langle \nabla f(x_t), v_t^{\text{away}} - x_t\rangle}{L\|v_t^{\text{away}} - x_t\|_2^2}, \gamma_{\max}\right\} & \text{if away step} \end{cases}$$

where $\gamma_{\max}$ is defined in the algorithm (see Algorithm 5 in Appendix C). Contrary to common belief, both strategies yield the same linear convergence rate; see Lacoste-Julien & Jaggi (2015) for AFW-ls and Theorem C.3 in the Appendix for AFW-L (Pedregosa et al., 2020). For BoostFW, we also ran a line search strategy to demonstrate that the speed-up really comes from the boosting procedure and not from being line search-free. Results further show that the line search-free strategy $\gamma_t \leftarrow \min\{\eta_t \|\nabla f(x_t)\| / (L\|g_t\|), 1\} = \min\{\langle -\nabla f(x_t), g_t\rangle / (L\|g_t\|^2), 1\}$ is very performant in CPU time. The line search-free strategy of DICG was not competitive in the experiments.

DICG is not applicable to optimization problems over the ℓ_1-ball

$$\min_{x\in\mathbb{R}^n} f(x) \quad \text{s.t. } \|x\|_1 \leqslant \tau, \tag{4}$$

however we can perform a change of variables $x_i = z_i - z_{n+i}$ and use the following reformulation over the simplex:

$$\min_{z \in \mathbb{R}^{2n}} f([z]_{1:n} - [z]_{n+1:2n}) \quad \text{s.t. } z \in \tau\Delta_{2n} \tag{5}$$

where $[z]_{1:n}$ and $[z]_{n+1:2n}$ denote the truncation to $\mathbb{R}^n$ of the first n entries and the last n entries of $z \in \mathbb{R}^{2n}$ respectively. Fact 4.1 formally states the equivalence between problems (4) and (5). A proof can be found in Appendix D.4.

Fact 4.1. *Consider $\mathbb{R}^n$ and let $\tau > 0$. Then $\mathcal{B}_1(\tau) = \{[z]_{1:n} - [z]_{n+1:2n} \mid z \in \tau\Delta_{2n}\}$.*

We implemented all the algorithms in Python using the same code framework for fair comparisons. In the case of synthetic data, we generated them from Gaussian distributions. We ran the experiments on a laptop under Linux Ubuntu 18.04 with Intel Core i7 3.5GHz CPU and 8GB RAM. Code is available at `https://github.com/cyrillewcombettes/boostfw`. In each experiment, we estimated the smoothness constant L of the (convex) objective function $f : \mathbb{R}^n \to \mathbb{R}$, i.e., the Lipschitz constant of the gradient function $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$, by sampling a few pairs of points $(x, y) \in \mathcal{C} \times \mathcal{C}$ and computing an upper bound on $\|\nabla f(y) - \nabla f(x)\|_2 / \|y - x\|_2$. Unless specified otherwise, we set $\delta \leftarrow 10^{-3}$ and $K \leftarrow +\infty$ in BoostFW. The role of K is only to cap the number of pursuit rounds per iteration when the FW oracle is particularly expensive (see Section 4.3).

4.1. Sparse signal recovery

Let $x^* \in \mathbb{R}^n$ be a signal which we want to recover as a sparse representation from observations $y = Ax^* + w$, where $A \in \mathbb{R}^{m \times n}$ and $w \sim \mathcal{N}(0, \sigma^2 I_m)$ is the noise in the measurements. The natural formulation of the problem is

$$\min_{x \in \mathbb{R}^n} \|y - Ax\|_2^2 \quad \text{s.t. } \|x\|_0 \leqslant \|x^*\|_0$$

but the ℓ_0-pseudo-norm $\|\cdot\|_0 : x \in \mathbb{R}^n \mapsto |\{i \in [\![1, n]\!] \mid [x]_i \neq 0\}|$ is nonconvex and renders the problem intractable in many situations (Natarajan, 1995). To remedy this, the ℓ_1-norm is often used as a convex surrogate and leads to the following lasso formulation (Tibshirani, 1996) of the problem:

$$\min_{x \in \mathbb{R}^n} \|y - Ax\|_2^2 \quad \text{s.t. } \|x\|_1 \leqslant \|x^*\|_1.$$

In order to compare to DICG, which is not applicable to this formulation, we ran all algorithms on the reformulation (5). We set $m = 200$, $n = 500$, $\sigma = 0.05$, and $\tau = \|x^*\|_1$. Since the objective function is quadratic, we can derive a closed-form solution to the line search and there is no need for AFW-L or BoostFW-L. The results are presented in Figure 4.

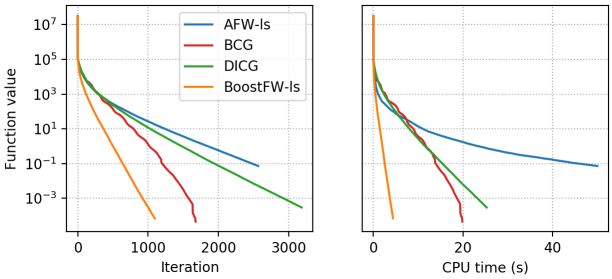

Figure 4. Sparse signal recovery.

4.2. Sparsity-constrained logistic regression

We consider the task of recognizing the handwritten digits 4 and 9 from the Gisette dataset (Guyon et al., 2005), available at `https://archive.ics.uci.edu/ml/datasets/Gisette`. The dataset includes a high number of distractor features with no predictive power. Hence, a sparsity-constrained logistic regression model is suited for the task. The sparsity-inducing constraint is realized via the ℓ_1-norm:

$$\min_{x \in \mathbb{R}^n} \frac{1}{m} \sum_{i=1}^{m} \ln(1 + \exp(-y_i a_i^\top x)) \quad \text{s.t. } \|x\|_1 \leqslant \tau$$

where $a_1, \ldots, a_m \in \mathbb{R}^n$ and $y \in \{-1, +1\}^m$. In order to compare to DICG, which is not applicable to this formulation, we ran all algorithms on the reformulation (5). We used $m = 2000$ samples and the number of features is $n = 5000$. We set $\tau = 10$, $L = 0.5$, and $\delta \leftarrow 10^{-4}$ in BoostFW. The results are presented in Figure 5. As expected, AFW-L and BoostFW-L converge faster in CPU time as they do not rely on line search, however they converge slower per iteration as each iteration provides less progress.

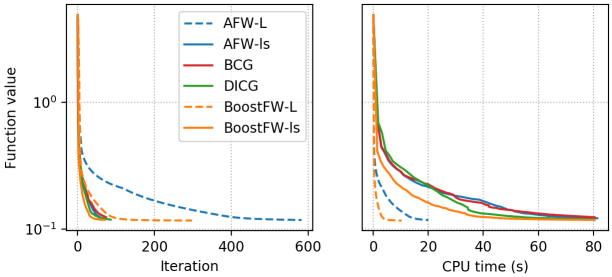

Figure 5. Sparse logistic regression on the Gisette dataset.

4.3. Traffic assignment

We consider the traffic assignment problem. The task is to assign vehicles on a traffic network in order to minimize congestion while satisfying travel demands. Let $\mathcal{A}$, $\mathcal{R}$, and $\mathcal{S}$ be the sets of links, routes, and origin-destination pairs respectively. For every pair $(i,j) \in \mathcal{S}$, let $\mathcal{R}_{i,j}$ and $d_{i,j}$ be the set of routes and the travel demand from i to j. Let x_a and t_a be the flow and the travel time on link $a \in \mathcal{A}$, and let y_r be the flow on route $r \in \mathcal{R}$. The Beckmann formulation of the problem (Beckmann et al., 1956), derived from the Wardrop equilibrium conditions (Wardrop, 1952), is

$$\begin{aligned} \min_{x \in \mathbb{R}^{|\mathcal{A}|}} \sum_{a \in \mathcal{A}} \int_0^{x_a} t_a(\xi)\,\mathrm{d}\xi & \qquad (6) \\ \text{s.t. } x_a = \sum_{r \in \mathcal{R}} \mathbb{1}_{\{a \in r\}} y_r & \quad a \in \mathcal{A} \\ \sum_{r \in \mathcal{R}_{i,j}} y_r = d_{i,j} & \quad (i,j) \in \mathcal{S} \\ y_r \geqslant 0 & \quad r \in \mathcal{R}_{i,j},\ (i,j) \in \mathcal{S}. \end{aligned}$$

A commonly used expression for the travel time t_a as a function of the flow x_a, developed by the Bureau of Public Records, is $t_a : x_a \in \mathbb{R}_+ \mapsto \tau_a(1 + 0.15(x_a/c_a)^4)$ where τ_a and c_a are the free-flow travel time and the capacity of the link. A linear minimization over the feasible region in (6) amounts to computing the shortest routes between all origin-destination pairs. Thus, the FW oracle is particularly expensive here so we capped the maximum number of rounds in BoostFW to $K \leftarrow 5$; see Figure 12 in Appendix A.2. We implemented the oracle using the function `all_pairs_dijkstra_path` from the Python package `networkx` (Hagberg et al., 2008). We created a directed acyclic graph with 500 nodes split into 20 layers of 25 nodes each, and randomly dropped links with probability 0.5 so $|\mathcal{A}| \approx 6000$ and $|\mathcal{S}| \approx 113000$. We set $d_{i,j} \sim \mathcal{U}([0,1])$ for every $(i,j) \in \mathcal{S}$. DICG is not applicable here and AFW-L and BoostFW-L were not competitive. The results are presented in Figure 6.

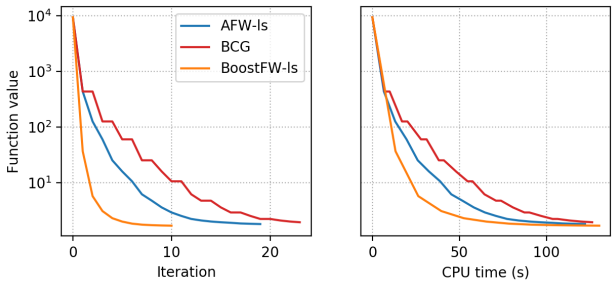

Figure 6. Traffic assignment.

4.4. Collaborative filtering

We consider the task of collaborative filtering on the MovieLens 100k dataset (Harper & Konstan, 2015), available at `https://grouplens.org/datasets/movielens/100k/`. The low-rank assumption on the solution and the approach of Mehta et al. (2007) lead to the following problem formulation:

$$\begin{aligned} \min_{X \in \mathbb{R}^{m \times n}} & \frac{1}{|\mathcal{I}|} \sum_{(i,j) \in \mathcal{I}} h_\rho(Y_{i,j} - X_{i,j}) \\ \text{s.t. } & \|X\|_{\text{nuc}} \leqslant \tau \end{aligned}$$

where h_ρ is the Huber loss with parameter $\rho > 0$ (Huber, 1964):

$$h_\rho : t \in \mathbb{R} \mapsto \begin{cases} t^2/2 & \text{if } |t| \leqslant \rho \\ \rho(|t| - \rho/2) & \text{if } |t| > \rho, \end{cases}$$

$Y \in \mathbb{R}^{m \times n}$ is the given matrix to complete, $\mathcal{I} \subseteq [\![1,m]\!] \times [\![1,n]\!]$ is the set of indices of observed entries in Y, and $\|\cdot\|_{\text{nuc}} : X \in \mathbb{R}^{m \times n} \mapsto \operatorname{tr}(\sqrt{X^\top X}) = \sum_{i=1}^{\min\{m,n\}} \sigma_i(X)$ is the nuclear norm and equals the sum of the singular vectors. It serves as a convex surrogate for the rank constraint (Fazel et al., 2001). Since

$$\begin{aligned} &\{X \in \mathbb{R}^{m \times n} \mid \|X\|_{\text{nuc}} = 1\} \\ &= \operatorname{conv}(\{uv^\top \mid u \in \mathbb{R}^m, v \in \mathbb{R}^n, \|u\|_2 = \|v\|_2 = 1\}), \end{aligned}$$

a linear minimization over the nuclear norm-ball of radius τ amounts to computing the top left and right singular vectors u and v of $-\nabla f(X_t)$ and to return $\tau uv^\top$. To this end, we used the function `svds` from the Python package `scipy.sparse.linalg` (Virtanen et al., 2020). We have $m = 943$, $n = 1682$, and $|\mathcal{I}| = 10^5$, and we set $\rho = 1$, $\tau = 5000$, and $L = 5 \cdot 10^{-6}$. DICG is not applicable here. The results are presented in Figure 7.

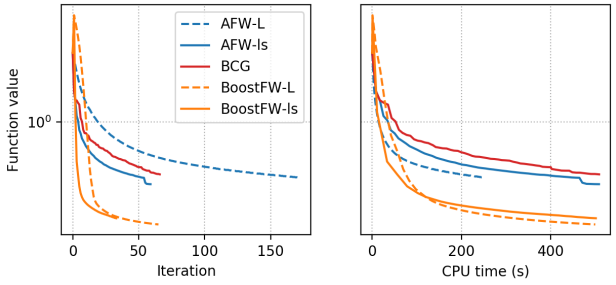

Figure 7. Collaborative filtering on the MovieLens 100k dataset.

The time limit here was set to 500 seconds but for AFW-L we reduced it to 250 seconds, else it raises a memory error on our machine shortly after. This is because AFW requires

storing the decomposition of the iterate onto $\mathcal{V}$. Note that BoostFW-ls converges faster in CPU time than AFW-L, although it relies on line search, and that BoostFW-L converges faster per iteration than the other methods although it does not rely on line search.

4.5. Video co-localization

We consider the task of video co-localization on the aeroplane class of the YouTube-Objects dataset (Prest et al., 2012), using the problem formulation of Joulin et al. (2014). The goal is to localize (with bounding boxes) the aeroplane object across the video frames. It consists in minimizing $f : x \in \mathbb{R}^{660} \mapsto x^\top Ax/2 + b^\top x$ over a flow polytope, where $A \in \mathbb{R}^{660\times 660}$, $b \in \mathbb{R}^{660}$, and the polytope each encode a part of the temporal consistency in the video frames. We obtained the data from `https://github.com/Simon-Lacoste-Julien/linearFW`. A linear minimization over the flow polytope amounts to computing a shortest path in the corresponding directed acyclic graph. We implemented the boosting procedure for DICG, which we labeled BoostDICG; see details in Appendix B. Since the objective function is quadratic, we can derive a closed-form solution to the line search and there is no need for AFW-L or BoostFW-L. We set $\delta \leftarrow 10^{-7}$ in BoostFW and $\delta \leftarrow 10^{-15}$ in BoostDICG. The results are presented in Figure 8.

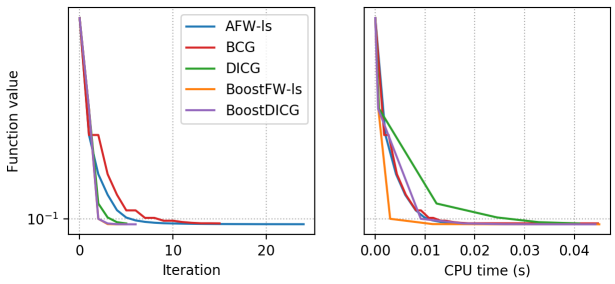

Figure 8. Video co-localization on the YouTube-Objects dataset.

All algorithms provide a similar level of performance in function value. In Garber & Meshi (2016), the algorithms are compared with respect to the duality gap $\max_{v\in\mathcal{V}}\langle\nabla f(x_t), x_t - v\rangle$ (Jaggi, 2013) on the same experiment. For completeness, we report a similar study in Figure 9. The boosting procedure applied to DICG produces very promising empirical results.

Appendix A.3 presents comparisons in duality gap for the other experiments. DICG converges faster than BoostFW in duality gap here (after closing it to 10^{-6} though), but it is not the case in the other experiments.

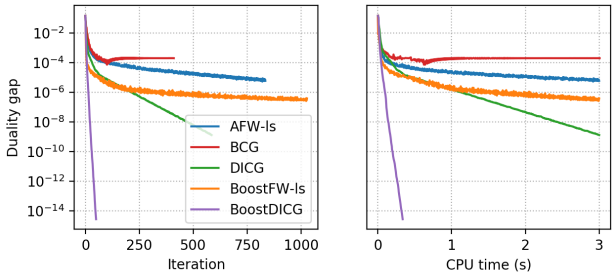

Figure 9. Video co-localization on the YouTube-Objects dataset.

5. Final remarks

We have proposed a new and intuitive method to speed up the Frank-Wolfe algorithm by descending in directions better aligned with those of the negative gradients $-\nabla f(x_t)$, all the while remaining projection-free. Our method does not need to maintain the decomposition of the iterates and can naturally be used to boost the performance of any Frank-Wolfe-style algorithm. Although the linear minimization oracle may be called multiple times per iteration, the progress obtained greatly overcomes this cost and leads to strong gains in performance. We demonstrated in a variety of experiments the computational advantage of our method both per iteration and in CPU time over the state-of-the-art. Furthermore, it does not require line search to produce strong performance in practice, which is particularly useful on instances where these are excessively expensive.

Future work may replace the gradient pursuit procedure with a faster conic optimization algorithm to potentially reduce the number of oracle calls. It could also be interesting to investigate how to make each oracle call cheaper via, e.g., *lazification* (Braun et al., 2017) or subsampling (Kerdreux et al., 2018). Lastly, we expect significant gains in performance when applying our approach to chase the gradient estimators in (non)convex stochastic Frank-Wolfe algorithms as well (Hazan & Luo, 2016; Xie et al., 2020).

Acknowledgments

Research reported in this paper was partially supported by NSF CAREER Award CMMI-1452463.

References

Bashiri, M. A. and Zhang, X. Decomposition-invariant conditional gradient for general polytopes with line search. In *Advances in Neural Information Processing Systems 30*, pp. 2690–2700. 2017.

Beckmann, M. J., McGuire, C. B., and Winsten, C. B. *Studies in the Economics of Transportation*. Yale University Press, 1956.

Bolte, J., Nguyen, T.-P., Peypouquet, J., and Suter, B. W. From

error bounds to the complexity of first-order descent methods for convex functions. *Mathematical Programming*, 165(2):471–507, 2017.

Braun, G., Pokutta, S., and Zink, D. Lazifying conditional gradient algorithms. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 566–575, 2017.

Braun, G., Pokutta, S., Tu, D., and Wright, S. Blended conditional gradients: the unconditioning of conditional gradients. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 735–743, 2019.

Canon, M. D. and Cullum, C. D. A tight upper bound on the rate of convergence of Frank-Wolfe algorithm. *SIAM Journal on Control*, 6(4):509–516, 1968.

Chen, J., Zhou, D., Yi, J., and Gu, Q. A Frank-Wolfe framework for efficient and effective adversarial attacks. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, pp. 3486–3494, 2020.

Fazel, M., Hindi, H., and Boyd, S. P. A rank minimization heuristic with application to minimum order system approximation. In *Proceedings of the American Control Conference*, pp. 4734–4739, 2001.

Frank, M. and Wolfe, P. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

Garber, D. and Hazan, E. Faster rates for the Frank-Wolfe method over strongly-convex sets. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 541–549, 2015.

Garber, D. and Hazan, E. A linearly convergent variant of the conditional gradient algorithm under strong convexity, with applications to online and stochastic optimization. *SIAM Journal on Optimization*, 26(3):1493–1528, 2016.

Garber, D. and Meshi, O. Linear-memory and decomposition-invariant linearly convergent conditional gradient algorithm for structured polytopes. In *Advances in Neural Information Processing Systems 29*, pp. 1001–1009. 2016.

Guélat, J. and Marcotte, P. Some comments on Wolfe's 'away step'. *Mathematical Programming*, 35(1):110–119, 1986.

Guyon, I., Gunn, S., Ben-Hur, A., and Dror, G. Result analysis of the NIPS 2003 feature selection challenge. In *Advances in Neural Information Processing Systems 17*, pp. 545–552. 2005.

Hagberg, A. A., Schult, D. A., and Swart, P. J. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference*, pp. 11 – 15, 2008.

Harper, F. M. and Konstan, J. A. The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 5(4):19:1–19:19, 2015.

Hazan, E. and Kale, S. Projection-free online learning. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

Hazan, E. and Luo, H. Variance-reduced and projection-free stochastic optimization. In *Proceedings of the 33rd International Conference on Machine Learning*, pp. 1263–1271, 2016.

Huber, P. J. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964.

Jaggi, M. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 427–435, 2013.

Joulin, A., Tang, K., and Fei-Fei, L. Efficient image and video co-localization with Frank-Wolfe algorithm. In *European Conference on Computer Vision*, pp. 253–268, 2014.

Karimi, H., Nutini, J., and Schmidt, M. Linear convergence of gradient and proximal-gradient methods under the Polyak-Łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 795–811, 2016.

Kerdreux, T., Pedregosa, F., and d'Aspremont, A. Frank-Wolfe with subsampling oracle. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 2591–2600, 2018.

Kurdyka, K. On gradients of functions definable in o-minimal structures. *Annales de l'Institut Fourier*, 48(3):769–783, 1998.

Lacoste-Julien, S. and Jaggi, M. On the global linear convergence of Frank-Wolfe optimization variants. In *Advances in Neural Information Processing Systems 28*, pp. 496–504. 2015.

Lacoste-Julien, S., Jaggi, M., Schmidt, M., and Pletscher, P. Block-coordinate Frank-Wolfe optimization for structural SVMs. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 53–61, 2013.

Lan, G. The complexity of large-scale convex programming under a linear optimization oracle. Technical report, Department of Industrial and Systems Engineering, University of Florida, 2013.

LeBlanc, L. J., Morlok, E. K., and Pierskalla, W. P. An efficient approach to solving the road network equilibrium traffic assignment problem. *Transportation Research*, 9(5):309–318, 1975.

Levitin, E. S. and Polyak, B. T. Constrained minimization methods. *USSR Computational Mathematics and Mathematical Physics*, 6(5):1–50, 1966.

Locatello, F., Tschannen, M., Rätsch, G., and Jaggi, M. Greedy algorithms for cone constrained optimization with convergence guarantees. In *Advances in Neural Information Processing Systems 30*, pp. 773–784. 2017.

Łojasiewicz, S. Une propriété topologique des sous-ensembles analytiques réels. In *Les Équations aux Dérivées Partielles*, 117, pp. 87–89. Colloques Internationaux du CNRS, 1963.

Mallat, S. G. and Zhang, Z. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12): 3397–3415, 1993.

Mehta, B., Hofmann, T., and Nejdl, W. Robust collaborative filtering. In *Proceedings of the 2007 ACM Conference on Recommender Systems*, pp. 49–56, 2007.

Natarajan, B. K. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2):227–234, 1995.

Pedregosa, F., Négiar, G., Askari, A., and Jaggi, M. Linearly convergent Frank-Wolfe with backtracking line-search. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*, pp. 1–10, 2020.

Ping, W., Liu, Q., and Ihler, A. T. Learning infinite RBMs with Frank-Wolfe. In *Advances in Neural Information Processing Systems 29*, pp. 3063–3071. 2016.

Polyak, B. T. Gradient methods for the minimisation of functionals. *USSR Computational Mathematics and Mathematical Physics*, 3(4):864–878, 1963.

Prest, A., Leistner, C., Civera, J., Schmid, C., and Ferrari, V. Learning object class detectors from weakly annotated video. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3282–3289, 2012.

Shalev-Shwartz, S., Gonen, A., and Shamir, O. Large-scale convex minimization with a low-rank constraint. In *Proceedings of the 28th International Conference on Machine Learning*, pp. 329–336, 2011.

Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors. Scipy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, 2020.

Wardrop, J. G. Some theoretical aspects of road traffic research. In *Proceedings of the Institute of Civil Engineers*, volume 1, pp. 325–378, 1952.

Wolfe, P. Convergence theory in nonlinear programming. In *Integer and Nonlinear Programming*, pp. 1–36. North-Holland, 1970.

Xie, J., Shen, Z., Zhang, C., Qian, H., and Wang, B. Efficient projection-free online methods with stochastic recursive gradient. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, pp. 6446–6453, 2020.