Enhancing Parameter-Free Frank Wolfe with an Extra Subproblem

Bingcong Li,[1] Lingda Wang,[2] Georgios B. Giannakis,[1] Zhizhen Zhao[2]
[1]University of Minnesota - Twin Cities
[2]University of Illinois at Urbana-Champaign
{lixx5599, georgios}@umn.edu, {lingdaw2, zhizhenz}@illinois.edu

Abstract

Aiming at convex optimization under structural constraints, this work introduces and analyzes a variant of the Frank Wolfe (FW) algorithm termed ExtraFW. The distinct feature of ExtraFW is the pair of gradients leveraged per iteration, thanks to which the decision variable is updated in a prediction-correction (PC) format. Relying on no problem dependent parameters in the step sizes, the convergence rate of ExtraFW for general convex problems is shown to be $\mathcal{O}(\frac{1}{k})$, which is optimal in the sense of matching the lower bound on the number of solved FW subproblems. However, the merit of ExtraFW is its faster rate $\mathcal{O}\big(\frac{1}{k^2}\big)$ on a class of machine learning problems. Compared with other parameter-free FW variants that have faster rates on the same problems, ExtraFW has improved rates and fine-grained analysis thanks to its PC update. Numerical tests on binary classification with different sparsity-promoting constraints demonstrate that the empirical performance of ExtraFW is significantly better than FW, and even faster than Nesterov's accelerated gradient on certain datasets. For matrix completion, ExtraFW enjoys smaller optimality gap, and lower rank than FW.

1 Introduction

The present work deals with efficient algorithms for solving the optimization problem

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

where f is a smooth convex function, while the constraint set $\mathcal{X}\subset\mathbb{R}^d$ is assumed to be convex and compact, and d is the dimension of the variable $\mathbf{x}$. Throughout we denote by $\mathbf{x}^*\in\mathcal{X}$ a minimizer of (1). For many machine learning and signal processing problems, the constraint set $\mathcal{X}$ can be structural but it is difficult or expensive to project onto. Examples include matrix completion in recommender systems (Freund, Grigas, and Mazumder 2017) and image reconstruction (Harchaoui, Juditsky, and Nemirovski 2015), whose constraint sets are nuclear norm ball and total-variation norm ball, respectively. The applicability of projected gradient descent (GD) (Nesterov 2004) and Nesterov's accelerated gradient (NAG) (Allen-Zhu and Orecchia 2014; Nesterov 2015) is thus limited by the computational barriers of projection, especially as d grows large.

An alternative to GD for solving (1) is the Frank Wolfe (FW) algorithm (Frank and Wolfe 1956; Jaggi 2013; Lacoste-Julien and Jaggi 2015), also known as the 'conditional gradient' method. FW circumvents the projection in GD by solving a subproblem with a *linear* loss per iteration. For a structural $\mathcal{X}$, such as the constraint sets mentioned earlier, it is possible to solve the subproblem either in closed form or through low-complexity numerical methods (Jaggi 2013; Garber and Hazan 2015), which saves computational cost relative to projection. In addition to matrix completion and image reconstruction, FW has been appreciated in several applications including structural SVM (Lacoste-Julien et al. 2013), video colocation (Joulin, Tang, and Fei-Fei 2014), optimal transport (Luise et al. 2019), and submodular optimization (Mokhtari, Hassani, and Karbasi 2018), to name a few.

Although FW has well documented merits, it exhibits slower convergence when compared to NAG. Specifically, FW satisfies $f(\mathbf{x}_k)-f(\mathbf{x}^*)=\mathcal{O}(\frac{1}{k})$, where the subscript k is iteration index. This convergence slowdown is confirmed by the lower bound, which indicates that the number of FW subproblems to solve in order to ensure $f(\mathbf{x}_k)-f(\mathbf{x}^*)\leq\epsilon$, is no less than $\mathcal{O}\big(\frac{1}{\epsilon}\big)$ (Lan 2013; Jaggi 2013). Thus, FW is a lower-bound-matching algorithm, in general. However, improved FW variants are possible either in empirical performance, or, in speedup rates for certain subclasses of problems. Next, we deal with these improved rates paying attention to whether implementation requires knowing parameters such as the smoothness constant or the diameter of $\mathcal{X}$.

Parameter-dependent FW with faster rates. This class of algorithms utilizes parameters that are obtained for different instances of f and $\mathcal{X}$. Depending on the needed parameters, these algorithms are further classified into: i) line search based FW; ii) shorter step size aided FW; and iii) conditional gradient sliding (CGS). Line search based FW relies on $f(\mathbf{x})$ evaluations, which renders inefficiency when acquisition of function values is costly. The vanilla FW with line search converges with rate $\mathcal{O}(\frac{1}{k})$ on general problems (Jaggi 2013). Jointly leveraging line search and 'away steps,' variants of FW converge linearly for strongly convex problems when $\mathcal{X}$ is a polytope (Guélat and Marcotte 1986; Lacoste-Julien and Jaggi 2015); see also (Pedregosa et al. 2018; Braun et al. 2018). To improve the memory efficiency of away steps, a variant is further developed in (Garber and Meshi 2016).

Shorter step sizes refer to those used in (Levitin and Polyak 1966; Garber and Hazan 2015), where the step size is obtained by minimizing a one-dimensional quadratic function over $[0, 1]$. Shorter step sizes require the smoothness parameter, which needs to be estimated for different loss functions. If $\mathcal{X}$ is strongly convex, and the optimal solution is at the boundary of $\mathcal{X}$, it is known that FW converges linearly (Levitin and Polyak 1966). For uniformly (and thus strongly) convex sets, faster rates are attained given that the optimal solution is at the boundary of $\mathcal{X}$ (Kerdreux, dAspremont, and Pokutta 2020). When both f and $\mathcal{X}$ are strongly convex, FW with shorter step size converges at a rate of $\mathcal{O}(\frac{1}{k^2})$, regardless of where the optimal solution resides (Garber and Hazan 2015). The last category is CGS, where both smoothness parameter and the diameter of $\mathcal{X}$ are necessary. In CGS, the subproblem of the original NAG that relies on projection is replaced by gradient sliding that solves a sequence of FW subproblems. A faster rate $\mathcal{O}(\frac{1}{k^2})$ is obtained at the price of: i) requiring at most $\mathcal{O}(k)$ FW subproblems in the kth iteration; and ii) an inefficient implementation since the NAG subproblem has to be solved up to a certain accuracy.

Parameter-free FW. The advantage of a parameter-free algorithm is its efficient implementation. Since no parameter is involved, there is no concern on the quality of parameter estimation. This also saves time and effort because the step sizes do not need tuning. Although implementation efficiency is ensured, theoretical guarantees are challenging to obtain. This is because $f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k)$ cannot be guaranteed without line search or shorter step sizes. Faster rates for parameter-free FW are rather limited in number, and most of existing parameter-free FW approaches rely on diminishing step sizes at the order of $\mathcal{O}(\frac{1}{k})$. For example, the behavior of FW when k is large and $\mathcal{X}$ is a polytope is investigated under strong assumptions on $f(\mathbf{x})$ to be twice differentiable and locally strongly convex around $\mathbf{x}^*$ (Bach 2020). AFW (Li et al. 2020) replaces the subproblem of NAG by a single FW subproblem, where constraint-specific faster rates are developed. Taking an active ℓ_2 norm ball constraint as an example, AFW guarantees a rate of $\mathcal{O}\big(\frac{\ln k}{k^2}\big)$. A natural question is whether the $\ln k$ in the numerator can be eliminated. In addition, although the implementation involves no parameter, the analysis of AFW relies on the value $\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$.

Aiming at parameter-free FW with faster rates (on certain constraints) that can bypass the limitations of AFW, the present work deals with the design and analysis of ExtraFW. The 'extra' in its name refers to the pair of gradients involved per iteration, whose merit is to enable a 'prediction-correction' (PC) type of update. Though the idea of using two gradients to perform PC updates originates from projection-based algorithms, such as ExtraGradient (Korpelevich 1976) and Mirror-Prox (Nemirovski 2004; Diakonikolas and Orecchia 2017; Kavis et al. 2019), leveraging PC updates in FW type algorithms for faster rates is novel.

Our contributions are summarized as follows.

- A new parameter-free FW variant, ExtraFW, is studied in this work. The distinct feature of ExtraFW is the adoption of two gradient evaluations per iteration to update the decision variable in a prediction-correction (PC) manner.
- It is shown that ExtraFW convergences with a rate of $\mathcal{O}(\frac{1}{k})$ for general problems. And for constraint sets including active ℓ_1, ℓ_2 and n-support norm balls, ExtraFW guarantees an accelerated rate $\mathcal{O}(\frac{1}{k^2})$.
- Unlike most of faster rates in FW literatures, ExtraFW is parameter-free, so that no problem dependent parameter is required. Compared with another parameter-free algorithm with faster rates, AFW (Li et al. 2020), introducing PC update in ExtraFW leads to several advantages: i) the convergence rate is improved by a factor of $\mathcal{O}(\ln k)$ on an ℓ_2 norm ball constraint; and ii) the analysis does not rely on the maximum value of $f(\mathbf{x})$ over $\mathcal{X}$.
- The efficiency of ExtraFW is corroborated on two benchmark machine learning tasks. The faster rate $\mathcal{O}(\frac{1}{k^2})$ is achieved on binary classification, evidenced by the possible improvement of ExtraFW over NAG on multiple sparsity-promoting constraint sets. For matrix completion, ExtraFW improves over AFW and FW in both optimality error and the rank of the solution.

Notation. Bold lowercase (uppercase) letters denote vectors (matrices); $\|\mathbf{x}\|$ stands for a norm of $\mathbf{x}$, with its dual norm written as $\|\mathbf{x}\|_*$; and $\langle \mathbf{x}, \mathbf{y}\rangle$ denotes the inner product of $\mathbf{x}$ and $\mathbf{y}$. We also define $x \wedge y := \min\{x, y\}$. Due to the page limitation, missing proofs can be found in Appendix.[1]

2 Preliminaries

This section reviews FW and AFW in order to illustrate the proposed algorithm in a principled manner. We first pinpoint the class of problems to focus on.

Assumption 1. *(Lipschitz Continuous Gradient.) The function $f : \mathcal{X} \to \mathbb{R}$ has L-Lipchitz continuous gradients; that is, $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_* \leq L\|\mathbf{x} - \mathbf{y}\|, \forall\, \mathbf{x}, \mathbf{y} \in \mathcal{X}$.*

Assumption 2. *(Convex Objective Function.) The function $f : \mathcal{X} \to \mathbb{R}$ is convex; that is, $f(\mathbf{y}) - f(\mathbf{x}) \geq \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x}\rangle, \forall\, \mathbf{x}, \mathbf{y} \in \mathcal{X}$.*

Assumption 3. *(Constraint Set.) The constraint set $\mathcal{X}$ is convex and compact with diameter D, that is, $\|\mathbf{x} - \mathbf{y}\| \leq D, \forall \mathbf{x}, \mathbf{y} \in \mathcal{X}$.*

Assumptions 1 – 3 are standard for FW type algorithms, and will be taken to hold true throughout. A blackbox optimization paradigm is considered in this work, where the objective function and constraint set can be accessed through oracles only. In particular, the first-order oracle (FO) and the linear minimization oracle (LMO) are needed.

Definition 1. *(FO.) The first-order oracle takes $\mathbf{x} \in \mathcal{X}$ as an input and returns its gradient $\nabla f(\mathbf{x})$.*

Definition 2. *(LMO.) The linear minimization oracle takes a vector $\mathbf{g} \in \mathbb{R}^d$ as an input and returns a minimizer of $\min_{\mathbf{x}\in\mathcal{X}} \langle \mathbf{g}, \mathbf{x}\rangle$.*

Except for gradients, problem dependent parameters such as function value, smoothness constant L, and constraint diameter D are not provided by FO and LMO. Hence, algorithms relying only on FO and LMO are parameter-free.

[1] http://arxiv.org/abs/2012.05284

Algorithm 1 FW (Frank and Wolfe 1956)

1: **Initialize:** $\mathbf{x}_0 \in \mathcal{X}$
2: **for** $k = 0, 1, \ldots, K-1$ **do**
3: $\quad \mathbf{v}_{k+1} = \arg\min_{\mathbf{x}\in\mathcal{X}} \langle \nabla f(\mathbf{x}_k), \mathbf{x} \rangle$
4: $\quad \mathbf{x}_{k+1} = (1-\delta_k)\mathbf{x}_k + \delta_k \mathbf{v}_{k+1}$
5: **end for**
6: **Return:** $\mathbf{x}_K$

Algorithm 2 AFW (Li et al. 2020)

1: **Initialize:** $\mathbf{x}_0 \in \mathcal{X}$, $\mathbf{g}_0 = \mathbf{0}$, and $\mathbf{v}_0 = \mathbf{x}_0$
2: **for** $k = 0, 1, \ldots, K-1$ **do**
3: $\quad \mathbf{y}_k = (1-\delta_k)\mathbf{x}_k + \delta_k \mathbf{v}_k$
4: $\quad \mathbf{g}_{k+1} = (1-\delta_k)\mathbf{g}_k + \delta_k \nabla f(\mathbf{y}_k)$
5: $\quad \mathbf{v}_{k+1} = \arg\min_{\mathbf{x}\in\mathcal{X}} \langle \mathbf{g}_{k+1}, \mathbf{x} \rangle$
6: $\quad \mathbf{x}_{k+1} = (1-\delta_k)\mathbf{x}_k + \delta_k \mathbf{v}_{k+1}$
7: **end for**
8: **Return:** $\mathbf{x}_K$

Next, we recap FW and AFW with parameter-free step sizes to gain more insights for the proposed algorithm.

FW recap. FW is summarized in Alg. 1. A subproblem with a linear loss, referred to also as an *FW step*, is solved per iteration via LMO. The FW step can be explained as finding a minimizer over $\mathcal{X}$ for the following supporting hyperplane of $f(\mathbf{x})$,

$$f(\mathbf{x}_k) + \langle \nabla f(\mathbf{x}_k), \mathbf{x} - \mathbf{x}_k \rangle. \tag{2}$$

Note that (2) is also a lower bound for $f(\mathbf{x})$ due to convexity. Upon obtaining $\mathbf{v}_{k+1}$ by minimizing (2) over $\mathcal{X}$, $\mathbf{x}_{k+1}$ is updated as a convex combination of $\mathbf{v}_{k+1}$ and $\mathbf{x}_k$ to eliminate the projection. The parameter-free step size is usually chosen as $\delta_k = \frac{2}{k+2}$. As for convergence, FW guarantees $f(\mathbf{x}_k) - f(\mathbf{x}^*) = \mathcal{O}(\frac{LD^2}{k})$.

AFW recap. As an FW variant, AFW in Alg. 2 relies on Nesterov momentum type update, that is, it uses an auxiliary variable $\mathbf{y}_k$ to estimate $\mathbf{x}_{k+1}$ and calculates the gradient $\nabla f(\mathbf{y}_k)$. If one writes $\mathbf{g}_{k+1}$ explicitly, $\mathbf{v}_{k+1}$ can be equivalently described as a minimizer over $\mathcal{X}$ of the hyperplane

$$\sum_{\tau=0}^{k} w_k^\tau \big[f(\mathbf{y}_\tau) + \langle \nabla f(\mathbf{y}_\tau), \mathbf{x} - \mathbf{y}_\tau \rangle \big] \tag{3}$$

where $w_k^\tau = \delta_\tau \prod_{j=\tau+1}^{k} (1-\delta_j)$ and $\sum_{\tau=0}^{k} w_k^\tau \approx 1$ (the sum depends on the choice of δ_0). Note that $f(\mathbf{y}_\tau) + \langle \nabla f(\mathbf{y}_\tau), \mathbf{x} - \mathbf{y}_\tau \rangle$ is a supporting hyperplane of $f(\mathbf{x})$ at $\mathbf{y}_\tau$, hence (3) is a lower bound for $f(\mathbf{x})$ constructed through a weighted average of supporting hyperplanes at $\{\mathbf{y}_\tau\}$. AFW converges at $\mathcal{O}\big(\frac{LD^2}{k}\big)$ on general problems. When the constraint set is an active ℓ_2 norm ball, AFW has a faster rate $\mathcal{O}\big(\frac{LD^2}{k} \wedge \frac{TLD^2 \ln k}{k^2}\big)$, where T depends on D. Writing this rate compactly as $\mathcal{O}\big(\frac{TLD^2 \ln k}{k^2}\big)$, it is observed that AFW achieves acceleration with the price of a worse dependence on other parameters hidden in T. However, even for the k-dependence, AFW is $\mathcal{O}(\ln k)$ times slower compared with other momentum based algorithms such as NAG. This slowdown is because that the lower bound (3) is constructed based on $\{\mathbf{y}_k\}$, which are estimated $\{\mathbf{x}_{k+1}\}$. We will show that relying on a lower bound constructed using $\{\mathbf{x}_{k+1}\}$ directly, it is possible to avoid this $\mathcal{O}(\ln k)$ slowdown.

Algorithm 3 ExtraFW

1: **Initialize:** $\mathbf{x}_0$, $\mathbf{g}_0 = \mathbf{0}$, and $\mathbf{v}_0 = \mathbf{x}_0$
2: **for** $k = 0, 1, \ldots, K-1$ **do**
3: $\quad \mathbf{y}_k = (1-\delta_k)\mathbf{x}_k + \delta_k \mathbf{v}_k$ ▷ prediction
4: $\quad \hat{\mathbf{g}}_{k+1} = (1-\delta_k)\mathbf{g}_k + \delta_k \nabla f(\mathbf{y}_k)$
5: $\quad \hat{\mathbf{v}}_{k+1} = \arg\min_{\mathbf{v}\in\mathcal{X}} \langle \hat{\mathbf{g}}_{k+1}, \mathbf{v} \rangle$
6: $\quad \mathbf{x}_{k+1} = (1-\delta_k)\mathbf{x}_k + \delta_k \hat{\mathbf{v}}_{k+1}$ ▷ correction
7: $\quad \mathbf{g}_{k+1} = (1-\delta_k)\mathbf{g}_k + \delta_k \nabla f(\mathbf{x}_{k+1})$
8: $\quad \mathbf{v}_{k+1} = \arg\min_{\mathbf{v}\in\mathcal{X}} \langle \mathbf{g}_{k+1}, \mathbf{v} \rangle$ ▷ extra FW step
9: **end for**
10: **Return:** $\mathbf{x}_K$

3 ExtraFW

This section introduces the main algorithm, ExtraFW, and establishes its constraint dependent faster rates.

3.1 Algorithm Design

ExtraFW is summarized in Alg. 3. Different from the vanilla FW and AFW, two FW steps (Lines 5 and 8 of Alg. 3) are required per iteration. Compared with other algorithms relying on two gradient evaluations, such as Mirror-Prox (Diakonikolas and Orecchia 2017; Kavis et al. 2019), ExtraFW reduces the computational burden of the projection. In addition, as an FW variant, ExtraFW can capture the properties such as sparsity or low rank promoted by the constraints more effectively through the update than those projection based algorithms. Detailed elaboration can be found in Section 4 and Appendix D. To facilitate comparison with FW and AFW, ExtraFW is explained through constructing lower bounds of $f(\mathbf{x})$ in a "prediction-correction (PC)" manner. The merits of the PC update compared with AFW are: i) the elimination of $\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$ in analysis; and ii) it improves the convergence rate on certain class of problems as we will see later.

Lower bound prediction. Similar to AFW, the auxiliary variable $\mathbf{y}_k$ in Line 3 of Alg. 3 can be viewed as an estimate of $\mathbf{x}_{k+1}$. The first gradient is evaluated at $\mathbf{y}_k$, and is incorporated into $\hat{\mathbf{g}}_{k+1}$, which is an estimate of the weighted average of $\{\nabla f(\mathbf{x})_\tau\}_{\tau=1}^{k+1}$. By expanding $\hat{\mathbf{g}}_{k+1}$, one can verify that $\hat{\mathbf{v}}_{k+1}$ can be obtained equivalently through minimizing the following weighted sum,

$$\sum_{\tau=0}^{k-1} w_k^\tau \Big[f(\mathbf{x}_{\tau+1}) + \langle \nabla f(\mathbf{x}_{\tau+1}), \mathbf{x} - \mathbf{x}_{\tau+1} \rangle \Big] + \delta_k \Big[f(\mathbf{y}_k) + \big\langle \nabla f(\mathbf{y}_k), \mathbf{x} - \mathbf{y}_k \big\rangle \Big], \tag{4}$$

where $w_\tau = \delta_\tau \prod_{j=\tau+1}^{k} (1-\delta_j)$ and $\sum_{\tau=0}^{k-1} w_\tau + \delta_k \approx 1$. Note that each term inside square brackets forms a supporting hyperplane of $f(\mathbf{x})$, hence (4) is an (approximated) lower bound of $f(\mathbf{x})$ because of convexity. As a prediction to $f(\mathbf{x}_{k+1}) + \langle \nabla f(\mathbf{x}_{k+1}), \mathbf{x} - \mathbf{x}_{k+1} \rangle$, the last bracket in (4) will be corrected once $\mathbf{x}_{k+1}$ is obtained.

Lower bound correction. The gradient $\nabla f(\mathbf{x}_{k+1})$ is used to obtain a weighted averaged gradients $\mathbf{g}_{k+1}$. By unrolling $\mathbf{g}_{k+1}$, one can find that $\mathbf{v}_{k+1}$ is a minimizer of the following (approximated) lower bound of $f(\mathbf{x})$

$$\sum_{\tau=0}^{k-1} w_k^\tau \Big[f(\mathbf{x}_{\tau+1}) + \big\langle \nabla f(\mathbf{x}_{\tau+1}), \mathbf{x} - \mathbf{x}_{\tau+1} \big\rangle \Big] + \delta_k \Big[f(\mathbf{x}_{k+1}) + \big\langle \nabla f(\mathbf{x}_{k+1}), \mathbf{x} - \mathbf{x}_{k+1} \big\rangle \Big]. \quad (5)$$

Comparing (4) and (5), we deduce that the terms in the last bracket of (4) are corrected to the true supporting hyperplane of $f(\mathbf{x})$ at $\mathbf{x}_{k+1}$. In sum, the FW steps in ExtraFW rely on lower bounds of $f(\mathbf{x})$ constructed in a weighted average manner similar to AFW. However, the key difference is that ExtraFW leverages the supporting hyperplanes at true variables $\{\mathbf{x}_k\}$ rather than the auxiliary ones $\{\mathbf{y}_k\}$ in AFW through a "correction" effected by (5). In the following subsections, we will show that the PC update in ExtraFW performs no worse than FW or AFW on general problems, while harnessing its own analytical merits on certain constraint sets.

3.2 Convergence of ExtraFW

We investigate the convergence of ExtraFW by considering the general case first. The analysis relies on the notion of estimate sequence (ES) introduced in (Nesterov 2004). An ES "estimates" f using a sequence of surrogate functions $\{\Phi_k(\mathbf{x})\}$ that are analytically tractable (e.g., being quadratic or linear). ES is formalized in the following definition.

Definition 3. *A tuple $\big(\{\Phi_k(\mathbf{x})\}_{k=0}^\infty, \{\lambda_k\}_{k=0}^\infty\big)$ is called an estimate sequence of function $f(\mathbf{x})$ if $\lim_{k\to\infty} \lambda_k = 0$ and for any $\mathbf{x} \in \mathcal{X}$ we have $\Phi_k(\mathbf{x}) \le (1-\lambda_k) f(\mathbf{x}) + \lambda_k \Phi_0(\mathbf{x})$.*

The construction of ES varies for different algorithms (see e.g., (Kulunchakov and Mairal 2019; Nesterov 2004; Lin, Mairal, and Harchaoui 2015; Li, Wang, and Giannakis 2020)). However, the reason to rely on the ES based analysis is similar, as summarized in the following lemma.

Lemma 1. *For $\big(\{\Phi_k(\mathbf{x})\}_{k=0}^\infty, \{\lambda_k\}_{k=0}^\infty\big)$ satisfying the definition of ES, if $f(\mathbf{x}_k) \le \min_{\mathbf{x}\in\mathcal{X}} \Phi_k(\mathbf{x}) + \xi_k, \forall k$, it is true that*

$$f(\mathbf{x}_k) - f(\mathbf{x}^*) \le \lambda_k \big(\Phi_0(\mathbf{x}^*) - f(\mathbf{x}^*)\big) + \xi_k, \forall\, k.$$

As shown in Lemma 1, λ_k and ξ_k jointly characterize the convergence rate of $f(\mathbf{x}_k)$. (Consider $\lambda_k = \mathcal{O}(\frac{1}{k})$ and $\xi_k = \mathcal{O}(\frac{1}{k})$ for an example.) Keeping Lemma 1 in mind, we construct *two* sequences of *linear* surrogate functions for analyzing ExtraFW, which highlight the differences of our analysis with existing ES based approaches

$$\Phi_0(\mathbf{x}) = \hat{\Phi}_0(\mathbf{x}) \equiv f(\mathbf{x}_0) \quad (6a)$$

$$\hat{\Phi}_{k+1}(\mathbf{x}) = (1-\delta_k)\Phi_k(\mathbf{x}) + \delta_k \big[f(\mathbf{y}_k) + \langle \nabla f(\mathbf{y}_k), \mathbf{x} - \mathbf{y}_k \rangle \big],\ \forall k \ge 0 \quad (6b)$$

$$\Phi_{k+1}(\mathbf{x}) = (1-\delta_k)\Phi_k(\mathbf{x}) + \big\langle \nabla f(\mathbf{x}_{k+1}), \mathbf{x} - \mathbf{x}_{k+1} \big\rangle \big],\ \forall k \ge 0. \quad (6c)$$

Clearly, both $\Phi_k(\mathbf{x})$ and $\hat{\Phi}_k(\mathbf{x})$ are linear in $\mathbf{x}$, in contrast to the quadratic surrogate functions adopted for analyzing NAG (Nesterov 2004). Such linear surrogate functions are constructed specifically for FW type algorithms taking advantage of the compact and convex constraint set. Next we show that (6) and proper $\{\lambda_k\}$ form two different ES of f.

Lemma 2. *If we choose $\lambda_0 = 1$, $\delta_k \in (0,1)$, and $\lambda_{k+1} = (1-\delta_k)\lambda_k\ \forall k \ge 0$, both $\big(\{\Phi_k(\mathbf{x})\}_{k=0}^\infty, \{\lambda_k\}_{k=0}^\infty\big)$ and $\big(\{\hat{\Phi}_k(\mathbf{x})\}_{k=0}^\infty, \{\lambda_k\}_{k=0}^\infty\big)$ satisfy the definition of ES.*

The key reason behind the construction of surrogate functions in (6) is that they are closely linked with the lower bounds (4) and (5) used in the FW steps, as stated in the next lemma.

Lemma 3. *Let $\mathbf{g}_0 = \mathbf{0}$, then it is true that $\mathbf{v}_k = \arg\min_{\mathbf{x}\in\mathcal{X}} \Phi_k(\mathbf{x})$ and $\hat{\mathbf{v}}_k = \arg\min_{\mathbf{x}\in\mathcal{X}} \hat{\Phi}_k(\mathbf{x})$.*

After relating the surrogate functions in (6) with ExtraFW, exploiting the analytical merits of the surrogate functions $\Phi_k(\mathbf{x})$ and $\hat{\Phi}_k(\mathbf{x})$, including being linear, next we show that $f(\mathbf{x}_k) \le \min_{\mathbf{x}\in\mathcal{X}} \Phi_k(\mathbf{x}) + \xi_k, \forall k$, which is the premise of Lemma 1.

Lemma 4. *Let $\xi_0 = 0$ and other parameters chosen the same as previous lemmas. Denote $\Phi_k^* := \Phi_k(\mathbf{v}_k)$ as the minimum value of $\Phi_k(\mathbf{x})$ over $\mathcal{X}$ (cf. Lemma 3), then ExtraFW guarantees that for any $k \ge 0$*

$$f(\mathbf{x}_k) \le \Phi_k^* + \xi_k, \text{ with } \xi_{k+1} = (1-\delta_k)\xi_k + \frac{3LD^2}{2}\delta_k^2.$$

Based on Lemma 4, the value of $f(\mathbf{x}_k)$ and Φ_k^* can be used to derive the stopping criterion if one does not want to preset the iteration number K. Further discussions are provided in Appendix A.6 due to space limitation.

Now we are ready to apply Lemma 1 to establish the convergence of ExtraFW.

Theorem 1. *Suppose that Assumptions 1, 2 and 3 are satisfied. Choosing $\delta_k = \frac{2}{k+3}$, and $\mathbf{g}_0 = \mathbf{0}$, ExtraFW in Alg. 3 guarantees*

$$f(\mathbf{x}_k) - f(\mathbf{x}^*) = \mathcal{O}\Big(\frac{LD^2}{k}\Big), \forall k.$$

This convergence rate of ExtraFW has the same order as AFW and FW. In addition, Theorem 1 translates into $\mathcal{O}(\frac{LD^2}{\epsilon})$ queries of LMO to ensure $f(\mathbf{x}_k) - f(\mathbf{x}^*) \le \epsilon$, which matches to the lower bound (Lan 2013; Jaggi 2013).

The obstacle for faster rates. As shown in the detailed proof, one needs to guarantee that either $\|\mathbf{v}_k - \hat{\mathbf{v}}_{k+1}\|^2$ or $\|\mathbf{v}_{k+1} - \hat{\mathbf{v}}_{k+1}\|^2$ is small enough to obtain a faster rate than Theorem 1. This is difficult in general because there could be multiple $\mathbf{v}_k$ and $\hat{\mathbf{v}}_k$ solving the FW steps. A simple example is to consider the ith entry $[\mathbf{g}_k]_i = 0$. The ith entry $[\mathbf{v}_k]_i$ can then be chosen arbitrarily as long as $\mathbf{v}_k \in \mathcal{X}$. The non-uniqueness of $\mathbf{v}_k$ prevents one from ensuring a small upper bound of $\|\mathbf{v}_k - \hat{\mathbf{v}}_{k+1}\|^2,\ \forall\, \mathbf{v}_k$. In spite of this, we will show that together with the structure on $\mathcal{X}$, ExtraFW can attain faster rates.

3.3 Acceleration of ExtraFW

In this subsection, we provide constraint-dependent accelerated rates of ExtraFW when $\mathcal{X}$ is some norm ball. Even for projection based algorithms, most of faster rates are obtained with step sizes depending on L (Nemirovski 2004; Diakonikolas and Orecchia 2017). Thus, faster rates for parameter-free algorithms are challenging to establish. An extra assumption is needed in this subsection.

Assumption 4. *The constraint is active, i.e.,* $\|\nabla f(\mathbf{x}^*)\|_2 \geq G > 0$.

It is natural to rely on the position of the optimal solution in FW type algorithms for analysis, and one can see this assumption also in (Levitin and Polyak 1966; Dunn 1979; Li et al. 2020; Kerdreux, dAspremont, and Pokutta 2020). For a number of machine learning tasks, Assumption 4 is rather mild. Relying on Lagrangian duality, it can be seen that problem (1) with a norm ball constraint is equivalent to the regularized formulation $\min_{\mathbf{x}} f(\mathbf{x}) + \gamma g(\mathbf{x})$, where $\gamma \geq 0$ is the Lagrange multiplier, and $g(\mathbf{x})$ denotes some norm. In view of this, Assumption 4 simply implies that $\gamma > 0$ in the equivalent regularized formulation, that is, the norm ball constraint plays the role of a regularizer. Given the prevalence of the regularized formulation in machine learning, it is worth investigating its equivalent constrained form (1) under Assumption 4.

Technically, the need behind Assumption 4 can be exemplified through a one-dimensional problem. Consider minimizing $f(x) = x^2$ over $\mathcal{X} = \{x | x \in [-1, 1]\}$. We clearly have $x^* = 0$ for which the constraint is inactive at the optimal solution. Recall a faster rate of ExtraFW requires $\|\hat{v}_{k+1} - v_{k+1}\|_2$ to be small. When x_k is close to $x^* = 0$, it can happen that $\hat{g}_{k+1} > 0$ and $g_{k+1} < 0$, leading to $\hat{v}_{k+1} = -1$ and $v_{k+1} = 1$. The faster rate is prevented by pushing v_{k+1} and $\hat{v}_{k+1}$ further apart from each other.

Next, we consider different instances of norm ball constraints as examples to the acceleration of ExtraFW. For simplicity of exposition, the intuition and technical details are discussed using an ℓ_2 norm ball constraint in the main test. Detailed analysis for ℓ_1 and n-support norm ball constraints are provided in Appendix.

ℓ_2 norm ball constraint. Consider $\mathcal{X} := \{\mathbf{x} | \|\mathbf{x}\|_2 \leq \frac{D}{2}\}$. In this case, $\mathbf{v}_{k+1}$ and $\hat{\mathbf{v}}_{k+1}$ admit closed-form solutions, taking $\mathbf{v}_{k+1}$ as an example,

$$\mathbf{v}_{k+1} = \arg\min_{\mathbf{x} \in \mathcal{X}} \langle \mathbf{g}_{k+1}, \mathbf{x} \rangle = -\frac{D}{2\|\mathbf{g}_{k+1}\|_2} \mathbf{g}_{k+1}. \tag{7}$$

We assume that when using $\mathbf{g}_{k+1}$ as the input to the LMO, the returned vector is given by (7). This is reasonable since it is what we usually implemented in practice. Though rarely happen, one can choose $\mathbf{v}_{k+1} = \hat{\mathbf{v}}_{k+1}$ to proceed if $\mathbf{g}_{k+1} = \mathbf{0}$. Similarly, we can simply set $\hat{\mathbf{v}}_{k+1} = \mathbf{v}_k$ if $\hat{\mathbf{g}}_{k+1} = \mathbf{0}$. The uniqueness of $\mathbf{v}_{k+1}$ is ensured by its closed-form solution, wiping out the obstacle for a faster rate.

Theorem 2. *Suppose that Assumptions 1, 2, 3 and 4 are satisfied, and $\mathcal{X}$ is an ℓ_2 norm ball. Choosing $\delta_k = \frac{2}{k+3}$, and $\mathbf{g}_0 = \mathbf{0}$, ExtraFW in Alg. 3 guarantees*

$$f(\mathbf{x}_k) - f(\mathbf{x}^*) = \mathcal{O}\left(\frac{LD^2}{k} \wedge \frac{LD^2 T}{k^2}\right), \forall k$$

where T is a constant depending only on L, G, and D.

Theorem 2 admits a couple of interpretations. By writing the rate compactly, ExtraFW achieves accelerated rate $\mathcal{O}\big(\frac{TLD^2}{k^2}\big), \forall k$ with a worse dependence on D compared to the vanilla FW. Or alternatively, the "asymptotic" performance at $k \geq T$ is strictly improved over the vanilla FW. It is worth mentioning that the choices of δ_k and $\mathbf{g}_0$ are not changed compared to Theorem 1 so that the parameter-free implementation is the same regardless whether accelerated. In other words, prior knowledge on whether Assumption 4 holds is not needed in practice. Compared with CGS, ExtraFW sacrifices the D dependence in the convergence rate to trade for i) the nonnecessity of the knowledge of L and D, and ii) ensuring two FW subproblems per iteration (whereas at most $\mathcal{O}(k)$ subproblems are needed in CGS). When comparing with AFW (Li et al. 2020), the convergence rate of ExtraFW is improved by a factor of $\mathcal{O}(\ln k)$, and the analysis does not rely on the constant $M := \max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x})$.

ℓ_1 norm ball constraint. For the sparsity-promoting constraint $\mathcal{X} := \{\mathbf{x} | \|\mathbf{x}\|_1 \leq R\}$, the FW steps can be solved in closed form too. Taking $\mathbf{v}_{k+1}$ as an example, we have

$$\mathbf{v}_{k+1} = R \cdot [0, \ldots, 0, -\mathrm{sgn}[\mathbf{g}_{k+1}]_i, 0, \ldots, 0]^\top \quad \text{with } \ i = \arg\max_j |[\mathbf{g}_{k+1}]_j|. \tag{8}$$

We show in Theorem 3 (see Appendix C.1) that when Assumption 4 holds and the set $\arg\max_j \big|[\nabla f(\mathbf{x}^*)]_j\big|$ has cardinality 1, a faster rate $\mathcal{O}(\frac{T_1 LD^2}{k^2})$ can be obtained with the constant T_1 depending on L, G, and D. The additional assumption here is known as *strict complementarity*, and has been adopted also in, e.g.,(Ding et al. 2020; Garber 2020).

n-support norm ball constraint. The n-support norm ball is a tighter relaxation of a sparsity prompting ℓ_0 norm ball combined with an ℓ_2 norm penalty compared with the ElasticNet (Zou and Hastie 2005). It is defined as $\mathcal{X} := \mathrm{conv}\{\mathbf{x} | \|\mathbf{x}\|_0 \leq n, \|\mathbf{x}\|_2 \leq R\}$, where $\mathrm{conv}\{\cdot\}$ denotes the convex hull (Argyriou, Foygel, and Srebro 2012). The closed-form solution of $\mathbf{v}_{k+1}$ is given by (Liu et al. 2016)

$$\mathbf{v}_{k+1} = -\frac{R}{\|\mathrm{top}_n(\mathbf{g}_{k+1})\|_2} \mathrm{top}_n(\mathbf{g}_{k+1}) \tag{9}$$

where $\mathrm{top}_n(\mathbf{g})$ denotes the truncated version of $\mathbf{g}$ with its top n (in magnitude) entries preserved. A faster rate $\mathcal{O}(\frac{T_2 LD^2}{k^2})$ is guaranteed by ExtraFW under Assumption 4, and a condition similar to strict complementarity (see Theorem 4 in the Appendix C.2). Again, the constant T_2 here depends on L, G, and D.

Other constraints. Note that the faster rates for ExtraFW are not limited to the exemplified constraint sets. In principle, if i) certain structure such as sparsity is promoted by the constraint set so that $\mathbf{x}^*$ is likely to lie on the boundary of $\mathcal{X}$; and ii) one can ensure the uniqueness of $\mathbf{v}_k$ through either a closed-form solution or a specific implementation manner, the acceleration of ExtraFW is achievable. Discussions for faster rates on a simplex $\mathcal{X}$ can be found in Appendix C.1. In addition, one can easily extend our results to the matrix case, where the constraint set is the Frobenius or the nuclear norm ball since they are ℓ_2 and ℓ_1 norms on the singular values of matrices, respectively.

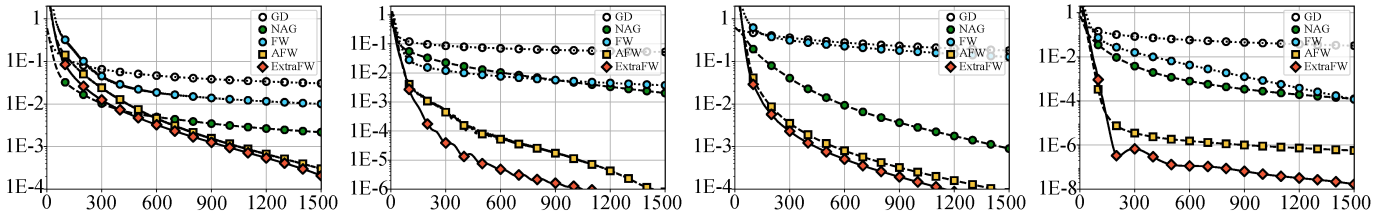

Figure 1: Performance of ExtraFW for binary classification with an ℓ_2 norm ball constraint on datasets: (a) *mnist*, (b) *w7a*, (c) *realsim*, and, (d) *mushroom*. In all plots, x-axis denotes the iteration number, and y-axis is $f(\mathbf{x}_k) - f(\mathbf{x}^*)$.

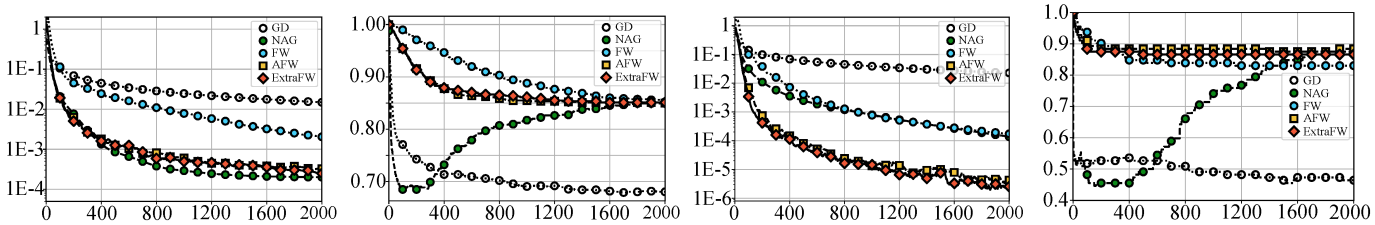

Figure 2: Performance of ExtraFW for binary classification with an ℓ_1 norm ball constraint: (a1) optimality error on *mnist*, (a2) solution sparsity on *mnist*, (b1) optimality error on *mushroom*, and, (b2) solution sparsity on *mushroom*. In all figures, x-axis denotes the iteration number.

4 Numerical Tests

This section deals with numerical tests of ExtraFW to showcase its effectiveness on different machine learning problems. Due to the space limitation, details of the datasets and implementation are deferred to Appendix D. For comparison, the benchmarked algorithms are chosen as: i) GD with standard step size $\frac{1}{L}$; ii) Nesterov accelerated gradient (NAG) with step sizes in (Allen-Zhu and Orecchia 2014); iii) FW with parameter-free step size $\frac{2}{k+2}$ (Jaggi 2013); and iv) AFW with step size $\frac{2}{k+3}$ (Li et al. 2020).

4.1 Binary Classification

We first investigate the performance of ExtraFW on binary classification using logistic regression. The constraints considered include: i) ℓ_2 norm ball for generalization merits; and, ii) ℓ_1 and n-support norm ball for promoting a sparse solution. The objective function is

$$f(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} \ln\left(1 + \exp(-b_i\langle \mathbf{a}_i, \mathbf{x}\rangle)\right) \qquad (10)$$

where $(\mathbf{a}_i, b_i)$ is the (feature, label) pair of datum i, and N is the number of data. Datasets *mnist*[2] and those from LIBSVM[3] are used in the numerical tests. Figures reporting test accuracy, and additional tests are postponed into Appendix.

ℓ_2 norm ball constraint. We start with $\mathcal{X} = \{\mathbf{x} | \|\mathbf{x}\|_2 \leq R\}$. The optimality error are plotted in Figure 1. On all tested datasets, ExtraFW outperforms AFW, NAG, FW and GD, demonstrating the $\mathcal{O}(\frac{1}{k^2})$ convergence rate established in Theorem 2. In addition, the simulation also suggests that T is in general small for logistic loss. On dataset *w7a* and *mushroom*, ExtraFW is significantly faster than AFW. All these observations jointly confirm the usefulness of the extra gradient and the PC update.

ℓ_1 norm ball constraint. Let $\mathcal{X} = \{\mathbf{x} | \|\mathbf{x}\|_1 \leq R\}$ be the constraint set to promote sparsity on the solution. Note that FW type updates directly guarantee that $\mathbf{x}_k$ has at most k non-zero entries when initialized at $\mathbf{x}_0 = \mathbf{0}$; see detailed discussions in Appendix D.2. In the simulation, R is tuned to obtain a solution that is almost as sparse as the dataset itself. The numerical results on datasets *mnist* and *mushroom* including both optimality error and the sparsity level of the solution can be found in Figure 2. On dataset *mnist*, ExtraFW slightly outperforms AFW but is not as fast as NAG. However, ExtraFW consistently finds solutions sparser than NAG. While on dataset *mushroom*, it can be seen that both AFW and ExtraFW outperform NAG, with ExtraFW slightly faster than AFW. And ExtraFW finds sparser solutions than NAG.

n-support norm ball constraint. Effective projection onto such a constraint is unknown yet and hence GD and NAG are not included in the test. The performance of ExtraFW can be found in Figure 3. On dataset *mnist*, both AFW and ExtraFW converge much faster than FW with ExtraFW slightly faster than AFW. However, FW trades the solution accuracy with its sparsity. On dataset *mushroom*, ExtraFW converges much faster than AFW and FW, while finding the sparsest solution.

4.2 Matrix Completion

We then consider matrix completion problems that are ubiquitous in recommender systems. Consider a matrix $\mathbf{A} \in$

[2]http://yann.lecun.com/exdb/mnist/

[3]https://www.csie.ntu.edu.tw/ cjlin/libsvm/

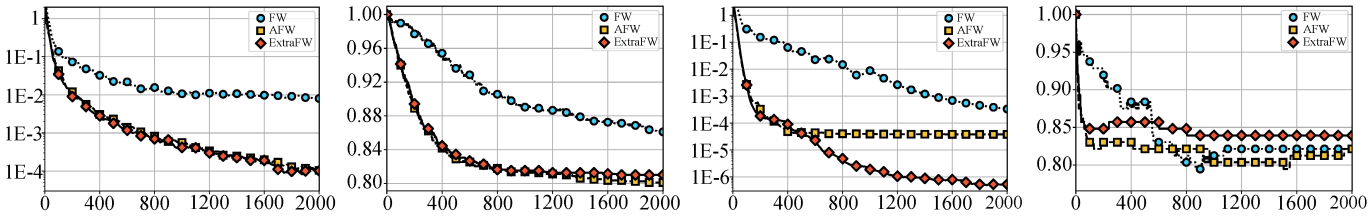

Figure 3: Performance of ExtraFW for binary classification with an n-support norm ball constraint: (a1) optimality error on *mnist*, (a2) solution sparsity on *mnist*, (b1) optimality error on *mushroom*, and, (b2) solution sparsity on *mushroom*. In all figures, x-axis denotes the iteration number.

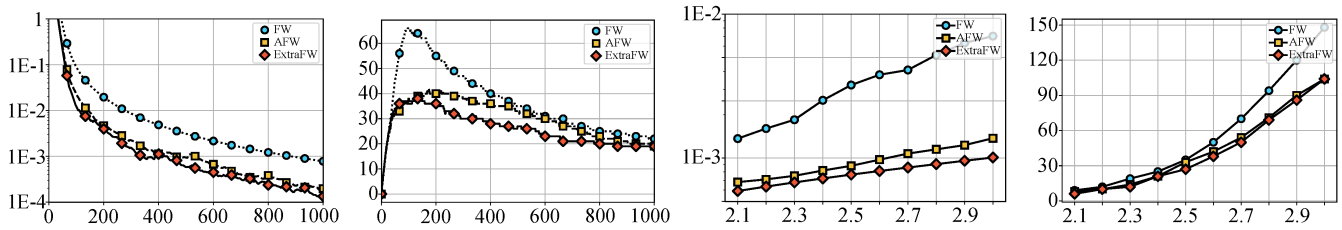

Figure 4: Performance of ExtraFW for matrix completion: (a) optimality vs k, (b) solution rank vs k, (c) optimality at $k = 500$ vs R, and, (d) solution rank at $k = 500$ vs R.

$\mathbb{R}^{m\times n}$ with partially observed entries, that is, entries A_{ij} for $(i,j) \in \mathcal{K}$ are known, where $\mathcal{K} \subset \{1,\dots,m\} \times \{1,\dots,n\}$. Note that the observed entries can also be contaminated by noise. The task is to predict the unobserved entries of $\mathbf{A}$. Although this problem can be approached in several ways, within the scope of recommender systems, a commonly adopted empirical observation is that $\mathbf{A}$ is low rank (Bennett and Lanning 2007; Bell and Koren 2007). Hence the objective boils down to

$$\min_{\mathbf{X}} \ \frac{1}{2} \sum_{(i,j)\in\mathcal{K}} (X_{ij} - A_{ij})^2 \qquad (11)$$
$$\text{s.t.} \ \ \|\mathbf{X}\|_{\text{nuc}} \le R$$

where $\|\cdot\|_{\text{nuc}}$ denotes the nuclear norm. Problem (11) is difficult to be solved via GD or NAG because projection onto a nuclear norm ball requires to perform SVD, which has complexity $\mathcal{O}\big(mn(m \wedge n)\big)$. On the contrary, FW and its variants are more suitable for (11) given the facts: i) Assumptions 1 – 3 are satisfied under nuclear norm (Freund, Grigas, and Mazumder 2017); ii) FW step can be solved easily with complexity at the same order as the number of nonzero entries; and iii) the update promotes low-rank solution directly (Freund, Grigas, and Mazumder 2017). More on ii) and iii) are discussed in Appendix D.3.

We test ExtraFW on a widely used dataset, *MovieLens100K*[4]. The experiments follow the same steps in (Freund, Grigas, and Mazumder 2017). The numerical performance of ExtraFW, AFW, and FW can be found in Figure 4. We plot the optimality error and rank versus k choosing $R = 2.5$ in Figures 4(a) and 4(b). It is observed that ExtraFW exhibits the best performance in terms of both optimality error and solution rank. In particular, ExtraFW roughly achieves 2.5x performance improvement compared with FW in terms of optimality error. We further compare the convergence of ExtraFW to AFW and FW at iteration $k = 500$ under different choices of R in Figures 4(c) and 4(d). ExtraFW still finds solutions with the lowest optimality error and rank. Moreover, the performance gap between ExtraFW and AFW increases with R, suggesting the inclined tendency of preferring ExtraFW over AFW and FW as R grows.

[4]https://grouplens.org/datasets/movielens/100k/

5 Conclusions

A new parameter-free FW variant, ExtraFW, is introduced and analyzed in this work. ExtraFW leverages two gradient evaluations per iteration to update in a "prediction-correction" manner. We show that ExtraFW converges at $\mathcal{O}(\frac{1}{k})$ on general problems, while achieving a faster rate $\mathcal{O}(\frac{TLD^2}{k^2})$ on certain types of constraint sets including active ℓ_1, ℓ_2 and n-support norm balls. Given the possibility of acceleration, ExtraFW is thus a competitive alternative to FW. The efficiency of ExtraFW is validated on tasks such as i) binary classification with different constraints, where ExtraFW can be even faster than NAG; and ii) matrix completion where ExtraFW finds solutions with lower optimality error and rank rapidly.

Acknowledgements

The authors would like to thank anonymous reviewers for their constructive feedback. BL and GG gratefully acknowledge the support from NSF grants 1711471, and 1901134. LW and ZZ are supported by Alfred P. Sloan Foundation.

References

Allen-Zhu, Z.; and Orecchia, L. 2014. Linear coupling: An ultimate unification of gradient and mirror descent. *arXiv preprint arXiv:1407.1537* .

Argyriou, A.; Foygel, R.; and Srebro, N. 2012. Sparse prediction with the k-support norm. In *Proc. Advances in Neural Info. Process. Syst.*, 1457–1465.

Bach, F. 2020. On the effectiveness of Richardson extrapolation in machine learning. *arXiv preprint arXiv:2002.02835* .

Bell, R. M.; and Koren, Y. 2007. Lessons from the Netflix prize challenge. *SiGKDD Explorations* 9(2): 75–79.

Bennett, J.; and Lanning, S. 2007. The Netflix prize. In *KDD cup and workshop*, 35. New York, NY, USA.

Braun, G.; Pokutta, S.; Tu, D.; and Wright, S. 2018. Blended conditional gradients: the unconditioning of conditional gradients. *arXiv preprint arXiv:1805.07311* .

Diakonikolas, J.; and Orecchia, L. 2017. Accelerated extra-gradient descent: A novel accelerated first-order method. *arXiv preprint arXiv:1706.04680* .

Ding, L.; Fei, Y.; Xu, Q.; and Yang, C. 2020. Spectral Frank-Wolfe algorithm: Strict complementarity and linear convergence. In *Proc. Intl. Conf. on Machine Learning*.

Dunn, J. C. 1979. Rates of convergence for conditional gradient algorithms near singular and nonsingular extremals. *SIAM Journal on Control and Optimization* 17(2): 187–211.

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

Freund, R. M.; Grigas, P.; and Mazumder, R. 2017. An extended Frank–Wolfe method with in-face directions, and its application to low-rank matrix completion. *SIAM Journal on Optimization* 27(1): 319–346.

Garber, D. 2020. Revisiting Frank-Wolfe for polytopes: Strict complementary and sparsity. *arXiv preprint arXiv:2006.00558* .

Garber, D.; and Hazan, E. 2015. Faster rates for the Frank-Wolfe method over strongly-convex sets. In *Proc. Intl. Conf. on Machine Learning*. Lille, France.

Garber, D.; and Meshi, O. 2016. Linear-memory and decomposition-invariant linearly convergent conditional gradient algorithm for structured polytopes. In *Proc. Advances in Neural Info. Process. Syst.*, 1001–1009.

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

Harchaoui, Z.; Juditsky, A.; and Nemirovski, A. 2015. Conditional gradient algorithms for norm-regularized smooth convex optimization. *Mathematical Programming* 152(1-2): 75–112.

Jaggi, M. 2013. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proc. Intl. Conf. on Machine Learning*, 427–435.

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

Kavis, A.; Levy, K. Y.; Bach, F.; and Cevher, V. 2019. UniXGrad: A universal, adaptive algorithm with optimal guarantees for constrained optimization. In *Proc. Advances in Neural Info. Process. Syst.*, 6257–6266.

Kerdreux, T.; dAspremont, A.; and Pokutta, S. 2020. Projection-free optimization on uniformly convex sets. *arXiv preprint arXiv:2004.11053* .

Korpelevich, G. 1976. The extragradient method for finding saddle points and other problems. *Matecon: Translations of Russian and East European Mathematical Economics* 12: 747–756.

Kulunchakov, A.; and Mairal, J. 2019. Estimate sequences for variance-reduced stochastic composite optimization. In *Proc. Intl. Conf. on Machine Learning*.

Lacoste-Julien, S.; and Jaggi, M. 2015. On the global linear convergence of Frank-Wolfe optimization variants. In *Proc. Advances in Neural Info. Process. Syst.*, 496–504. Atlanta, USA.

Lacoste-Julien, S.; Jaggi, M.; Schmidt, M. W.; and Pletscher, P. 2013. Block-coordinate Frank-Wolfe optimization for structural SVMs. In *Proc. Intl. Conf. on Machine Learning*, 53–61.

Lan, G. 2013. The complexity of large-scale convex programming under a linear optimization oracle. *arXiv:1309.5550* .

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

Li, B.; Coutino, M.; Giannakis, G. B.; and Leus, G. 2020. How does momentum help Frank Wolfe? *arXiv preprint arXiv:1908.09345* .

Li, B.; Wang, L.; and Giannakis, G. B. 2020. Almost tune-free variance reduction. In *Proc. Intl. Conf. on Machine Learning*.

Lin, H.; Mairal, J.; and Harchaoui, Z. 2015. A universal catalyst for first-order optimization. In *Proc. Advances in Neural Info. Process. Syst.*, 3384–3392. Montreal, Canada.

Liu, B.; Yuan, X.-T.; Zhang, S.; Liu, Q.; and Metaxas, D. N. 2016. Efficient k-support-norm regularized minimization via fully corrective Frank-Wolfe method. In *Proc. Intl. Joint Conf. on Artifical Intelligence*, 1760–1766.

Luise, G.; Salzo, S.; Pontil, M.; and Ciliberto, C. 2019. Sinkhorn barycenters with free support via Frank-Wolfe algorithm. In *Proc. Advances in Neural Info. Process. Syst.*, 9318–9329.

Mokhtari, A.; Hassani, H.; and Karbasi, A. 2018. Stochastic conditional gradient methods: From convex minimization to submodular maximization. *arXiv preprint arXiv:1804.09554* .

Nemirovski, A. 2004. Prox-method with rate of convergence $\mathcal{O}(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization* 15(1): 229–251.

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

Nesterov, Y. 2015. Universal gradient methods for convex optimization problems. *Mathematical Programming* 152(1-2): 381–404.

Pedregosa, F.; Askari, A.; Negiar, G.; and Jaggi, M. 2018. Step-size adaptivity in projection-free optimization. *arXiv preprint arXiv:1806.05123* .

Zou, H.; and Hastie, T. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: series B (Statistical Methodology)* 67(2): 301–320.