Stochastic Frank-Wolfe for Constrained Finite-Sum Minimization

Geoffrey Négiar [1] **Gideon Dresdner** [2] **Alicia Yi-Ting Tsai** [1]
Laurent El Ghaoui [1 3] **Francesco Locatello** [2 4] **Robert M. Freund** [5] **Fabian Pedregosa** [6]

Abstract

We propose a novel Stochastic Frank-Wolfe (a.k.a. conditional gradient) algorithm for constrained smooth finite-sum minimization with a generalized linear prediction/structure. This class of problems includes empirical risk minimization with sparse, low-rank, or other structured constraints. The proposed method is simple to implement, does not require step-size tuning, and has a constant per-iteration cost that is independent of the dataset size. Furthermore, as a byproduct of the method we obtain a stochastic estimator of the Frank-Wolfe gap that can be used as a stopping criterion. Depending on the setting, the proposed method matches or improves on the best computational guarantees for Stochastic Frank-Wolfe algorithms. Benchmarks on several datasets highlight different regimes in which the proposed method exhibits a faster empirical convergence than related methods. Finally, we provide an implementation of all considered methods in an open-source package.

1 Introduction

We consider constrained finite-sum optimization problems of the form

$$\underset{\boldsymbol{w}\in\mathcal{C}}{\text{minimize}} \ \frac{1}{n}\sum_{i=1}^{n} f_i\left(\boldsymbol{x}_i^\top \boldsymbol{w}\right), \qquad \text{(OPT)}$$

where $\mathcal{C}$ is a compact and convex set and $\boldsymbol{X} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)^\top \in \mathbb{R}^{n\times d}$ is a data matrix, with n samples and d features. This template includes several problems of interest, such as constrained empirical risk minimization. The LASSO (Tibshirani, 1996) may be written in this form, where $f_i(\boldsymbol{x}_i^\top \boldsymbol{w}) = \frac{1}{2}(\boldsymbol{x}_i^\top \boldsymbol{w} - y_i)^2$ and $\mathcal{C} = \{\boldsymbol{w} : \|\boldsymbol{w}\|_1 \leq \lambda\}$ for some parameter λ. We focus on the case where the f_is are differentiable with L-Lipschitz derivative, and study the convex and non-convex cases.

Table 1. Worst-case convergence rates for the function suboptimality after t iterations, for a dataset with n samples. $\kappa \leq n$ and can be much smaller than n for datasets of interest. κ is introduced in Section 5.

Related Work	Convex	Non-Convex
Frank & Wolfe (1956)	$\mathcal{O}(n/t)$	$\mathcal{O}\left(n/\sqrt{t}\right)$
Mokhtari et al. (2018)	$\mathcal{O}\left(1/\sqrt[3]{t}\right)$	✗
Lu & Freund (2018)	$\mathcal{O}(n/t)$	✗
This work	$\mathcal{O}(\kappa/t)$	$\to 0$

The classical Frank-Wolfe (FW) or Conditional Gradient algorithm (Frank & Wolfe, 1956; Levitin & Polyak, 1966; Demyanov & Rubinov, 1967) is an algorithm for constrained optimization. Contrary to other projection-based constrained optimization algorithms, such as Projected Gradient Descent, it relies on a Linear Minimization Oracle (LMO) over the constraint set $\mathcal{C}$, rather than a Quadratic Minimization Oracle (the projection subroutine). For certain constraint sets such as the trace norm or most ℓ_p balls, the LMO can be computed more efficiently than the projection subroutine. Recently, the Frank-Wolfe algorithm has garnered much attention in the machine learning community where polytope constraints and sparsity are of large interest, e.g. Jaggi (2013); Lacoste-Julien & Jaggi (2015); Locatello et al. (2017).

In the unconstrained setting, stochastic variance-reduced methods (Shalev-Shwartz & Zhang, 2013; Schmidt et al., 2013; Hofmann et al., 2015) exhibit the same iteration complexity as full gradient (non-stochastic) methods, while reaching much smaller per-iteration complexity, usually at some (small) additional memory cost. This work takes a step in the direction of designing such a method for Frank-Wolfe type algorithms, which remains an important open problem.

[1]Berkeley AI Research, University of California, Berkeley, CA, USA [2]Department of Computer Science, ETH Zurich, Switzerland [3]SumUp Analytics [4]Max-Planck Institute for Intelligent Systems, Tübingen, Germany [5]MIT Sloan School of Management [6]Google Research. Correspondence to: Geoffrey Négiar <geoffrey_negiar@berkeley.edu>.

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

Our main contributions are:

1. A **constant batch-size Stochastic Frank-Wolfe (SFW) algorithm for finite sums with linear prediction**. We describe the method in Section 2 and discuss its computational and memory cost.
2. A **non-asymptotic rate analysis on smooth and convex objectives**. The suboptimality of the SFW algorithm after t iterations can be bounded as $\mathcal{O}(\kappa/t)$, where κ is a data-dependent constant we will discuss later. It is upper bounded by the sample-size n but, depending on the setting, can be potentially much smaller.
3. An **asymptotic analysis for non-convex objectives**. We prove that SFW converges to a stationary point for smooth but potentially non-convex functions. This is the first stochastic FW variant that has convergence guarantees in this setting of large practical interest.

Finally, we compare the SFW algorithm with other stochastic Frank-Wolfe algorithms amenable to constant batch size on different machine learning tasks. These experiments show that the proposed method converges at least as fast as previous work, and notably faster on several such instances.

1.1 Related Work

We split existing stochastic FW algorithm into two categories: methods with increasing batch size and methods with constant batch size.

Increasing batch size Stochastic Frank-Wolfe. This variant allows the number of gradient evaluations to grow with the iteration number (Goldfarb et al., 2017; Hazan & Luo, 2016; Reddi et al., 2016). Because of the growing number of gradient evaluations, these methods converge towards a deterministic full gradient FW algorithm and so asymptotically share their computational requirements. In this work we will instead be interested in *constant batch-size* methods, in which the number of gradient evaluations does not increase with the iteration number. See Hazan & Luo (2016) for a detailed comparison of assumptions and complexities for Stochastic Frank-Wolfe methods with increasing batch sizes, in terms of both iterations and gradient calls.

Constant batch size Stochastic Frank-Wolfe. These methods use a constant batch size b, which is chosen by the user as a hyperparameter. In the convex and smooth setting, Mokhtari et al. (2018) and Locatello et al. (2019) reach $\mathcal{O}\left(1/\sqrt[3]{t}\right)$ convergence rates. The rate of Locatello et al. (2019) further holds for non-smooth and non-Lipschitz objectives. Zhang et al. (2019) requires second order knowledge of the objective. Lu & Freund (2018) proves convergence for an averaged iterate in $\mathcal{O}(n/t)$ with n the number of samples in the dataset. Let us assume for simplicity that we use unit batch size. Since each iteration involves only one data point, the per-iteration complexity of their method reduces by a factor of n the per-iteration complexity of full-gradient method. On the other hand, the method proposed in this work loses this factor in the rate in number of iterations, reaching the same overall complexity as the deterministic full gradient method. Depending on the use-case (large or small datasets), each of the rates reported in Lu & Freund (2018) and Mokhtari et al. (2018) can have an advantage over the other. In favorable cases, the rate of convergence achieved by our method is nearly independent of the number of samples in the dataset. In these cases, our method is therefore faster than both. In the worst case, it matches the $\mathcal{O}(n/t)$ bound (Lu & Freund, 2018).

1.2 Notation

Throughout the paper we denote vectors in lowercase boldface letters ($\boldsymbol{w}$), matrices in uppercase boldface letters ($\boldsymbol{X}$), and sets in calligraphic letters (e.g., $\mathcal{C}$). We say a function f is L-smooth in the norm $\|\cdot\|$ if it is differentiable and its gradient is L-Lipschitz continuous with respect to $\|\cdot\|$, that is, if it verifies $\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|_* \leq L\|\boldsymbol{x} - \boldsymbol{y}\|$ for all $\boldsymbol{x}, \boldsymbol{y}$ in the domain (where $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$). For a one dimensional function f, this reduces to $|f'(z) - f'(z')| \leq L|z - z'|$ for all $z,\ z'$ in the domain. For the time dependent vector $\boldsymbol{u}_t$, we denote by $\boldsymbol{u}_t^{(i)}$ its i-th coordinate.

We distinguish $\mathbb{E}$, the full expectation taken with respect to all the randomness in the system, from $\mathbb{E}_t$, the conditional expectation with respect to the random index sampled at iteration t, conditioned on all randomness up to iteration t.

Finally, $\mathrm{LMO}(\boldsymbol{u})$ returns an arbitrary element in $\arg\min_{\boldsymbol{s}\in\mathcal{C}} \langle \boldsymbol{s}, \boldsymbol{u}\rangle$.

2 Methods

2.1 A Primal-Dual View on Frank-Wolfe

In this subsection, we present the Frank-Wolfe algorithm as an alternating optimization scheme on a saddle-point problem. This point of view motivates the design of the proposed SFW algorithm. This perspective is similar to the two player game point of view of Abernethy & Wang (2017); Abernethy et al. (2018), which we express using convex conjugacy. We suppose here that f is closed, convex and differentiable.

Let us rewrite our initial problem (OPT) in the equivalent unconstrained formulation

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}}\ f(\boldsymbol{X}\boldsymbol{w}) + \imath_{\mathcal{C}}(\boldsymbol{w})\,, \tag{1}$$

where $\imath_{\mathcal{C}}$ is the indicator function of $\mathcal{C}$: it is 0 over $\mathcal{C}$ and

$+\infty$ outside of $\mathcal{C}$.

We denote by f^* the convex conjugate of f, that is, $f^*(\boldsymbol{\alpha}) \stackrel{\text{def}}{=} \max_{\boldsymbol{w}} \langle \boldsymbol{\alpha}, \boldsymbol{w} \rangle - f(\boldsymbol{w})$. Whenever f is closed and convex, it is known that $f = (f^*)^*$, and so we can write $f(\boldsymbol{X}\boldsymbol{w}) = \max_{\boldsymbol{\alpha}} \{-f^*(\boldsymbol{\alpha}) + \langle \boldsymbol{X}\boldsymbol{w}, \boldsymbol{\alpha} \rangle\}$. Plugging this identity into the previous equation, we arrive at a saddle-point reformulation of the original problem:

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \left\{ \mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}) \stackrel{\text{def}}{=} -f^*(\boldsymbol{\alpha}) + \imath_{\mathcal{C}}(\boldsymbol{w}) + \langle \boldsymbol{X}\boldsymbol{w}, \boldsymbol{\alpha} \rangle \right\}. \tag{2}$$

This reformulation allows to derive the Frank-Wolfe algorithm as an alternating optimization method on this saddle-point reformulation. To distinguish the algorithm in this section from the stochastic algorithm we propose, we denote the iterates in this section by $\bar{\boldsymbol{\alpha}}_t, \bar{\boldsymbol{w}}_t$.

The first step of the Frank-Wolfe algorithm is to compute the gradient of the objective at the current iterate. In the saddle-point formulation, this corresponds to maximizing over the dual variable $\boldsymbol{\alpha}$ at step t:

$$\begin{aligned} \bar{\boldsymbol{\alpha}}_t &\in \arg\max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \{\mathcal{L}(\bar{\boldsymbol{w}}_{t-1}, \boldsymbol{\alpha}) = -f^*(\boldsymbol{\alpha}) + \langle \boldsymbol{X}\bar{\boldsymbol{w}}_{t-1}, \boldsymbol{\alpha} \rangle\} \\ &\iff \bar{\boldsymbol{\alpha}}_t = \nabla f(\boldsymbol{X}\bar{\boldsymbol{w}}_{t-1}). \end{aligned} \tag{3}$$

Then, the LMO step corresponds to fixing the dual variable and minimizing over the primal one $\boldsymbol{w}$. This gives

$$\begin{aligned} \bar{\boldsymbol{s}}_t &\in \arg\min_{\boldsymbol{w} \in \mathbb{R}^d} \left\{ \mathcal{L}(\boldsymbol{w}, \bar{\boldsymbol{\alpha}}_t) = \imath_{\mathcal{C}}(\boldsymbol{w}) + \langle \boldsymbol{w}, \boldsymbol{X}^T \bar{\boldsymbol{\alpha}}_t \rangle \right\} \\ &\iff \bar{\boldsymbol{s}}_t = \mathrm{LMO}(\boldsymbol{X}^\top \bar{\boldsymbol{\alpha}}_t). \end{aligned} \tag{4}$$

Note that from the definition of the LMO, $\bar{\boldsymbol{s}}_t$ can always be chosen as an extreme point of the constraint set $\mathcal{C}$. We then update our iterate using the convex combination

$$\bar{\boldsymbol{w}}_t = (1 - \gamma_t)\bar{\boldsymbol{w}}_{t-1} + \gamma_t \bar{\boldsymbol{s}}_t, \tag{5}$$

where γ_t is a step-size to be chosen. These updates determine the Frank-Wolfe algorithm.

2.2 The Stochastic Frank-Wolfe Algorithm

We now consider a variant in which we replace the exact minimization of the dual variable (3) by a minimization over a single coordinate, chosen uniformly at random.

Let us define the function f from $\mathbb{R}^n$ to $\mathbb{R}$ as $f(\boldsymbol{\theta}) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n f_i(\boldsymbol{\theta}_i)$. We can write our original optimization problem as an optimization over $\boldsymbol{w} \in \mathcal{C}$ of $f(\boldsymbol{X}\boldsymbol{w})$. Still alternating between the primal and the dual problems, we replace maximization over the full vector $\boldsymbol{\alpha}$ in (3) with optimization along the coordinate i only. We obtain the update $\boldsymbol{\alpha}_t^{(i)} = \frac{1}{n} f_i'(\boldsymbol{x}_i^\top \boldsymbol{w}_{t-1})$. Doing so changes the cost per-iteration from $\mathcal{O}(nd)$ to $\mathcal{O}(d)$, and yields Algorithm 1.

Algorithm 1 Stochastic Frank-Wolfe

Initialization: $\boldsymbol{w}_0 \in \mathcal{C}$, $\boldsymbol{\alpha}_0 \in \mathbb{R}^n$, $\boldsymbol{r}_0 = \boldsymbol{X}^\top \boldsymbol{\alpha}_0$
for $t = 1, 2, \ldots,$ **do**
  Sample $i \in \{1, \ldots, n\}$ uniformly at random.
  Update $\boldsymbol{\alpha}_t^{(i)} = \frac{1}{n} f_i'(\boldsymbol{x}_i^\top \boldsymbol{w}_{t-1})$
  Update $\boldsymbol{\alpha}_t^{(j)} = \boldsymbol{\alpha}_{t-1}^j$, $j \neq i$
  $\boldsymbol{r}_t = \boldsymbol{r}_{t-1} + (\boldsymbol{\alpha}_t^{(i)} - \boldsymbol{\alpha}_{t-1}^{(i)})\boldsymbol{x}_i$
  $\boldsymbol{s}_t = \mathrm{LMO}(\boldsymbol{r}_t)$
  $\boldsymbol{w}_t = \boldsymbol{w}_{t-1} + \frac{2}{t+2}(\boldsymbol{s}_t - \boldsymbol{w}_{t-1})$
end for

We now describe our main contribution, Algorithm 1 (SFW) above. It follows the classical Frank-Wolfe algorithm, but replaces the gradient with a stochastic estimate of the gradient.

Throughout Algorithm 1, we maintain the following iterates:

- the iterate $\boldsymbol{w}_t$,
- the stochastic estimator of $\nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1})$ denoted by $\boldsymbol{\alpha}_t \in \mathbb{R}^n$,
- the stochastic estimator of the full gradient of our loss $\boldsymbol{X}^\top \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1})$, denoted by $\boldsymbol{r}_t \in \mathbb{R}^d$.

Algorithm. At the beginning of iteration t, we have access to $\boldsymbol{\alpha}_{t-1}$, $\boldsymbol{r}_{t-1}$ and to the iterate $\boldsymbol{w}_{t-1}$.

Thus equipped, we sample an index i uniformly at random over $\{1, \ldots, n\}$. We then compute the gradient of our loss function for that datapoint, on our iterate, yielding $[\nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1})]_i = \frac{1}{n} f_i'(\boldsymbol{x}_i^\top \boldsymbol{w}_{t-1})$. We update the stochastic gradient estimator $\boldsymbol{\alpha}_t$ by refreshing the contribution of the i-th datapoint and leaving the other coordinates untouched.

Remark 1. *Coordinate j of our estimator $\boldsymbol{\alpha}_t$ contains the latest sampled one-dimensional derivative of $\frac{1}{n} f_j$.*

To get $\boldsymbol{r}_t$, we do the same, removing the previous contribution of the i-th datapoint, and adding the refreshed contribution. This allows us not to store the full data-matrix in memory.

The rest of the algorithm continues as the deterministic Frank-Wolfe algorithm from the previous subsection: we find the update direction from $\boldsymbol{s}_t = \mathrm{LMO}(\boldsymbol{r}_t)$, and we update our iterate using a convex combination of the previous iterate $\boldsymbol{w}_{t-1}$ and $\boldsymbol{s}_t$, whereby our new iterate is feasible.

Remark 2. *Our algorithm requires to keep track of the $\boldsymbol{\alpha}_t$ vector and amounts to keeping one scalar per sample*

in memory. Our method requires the same small memory caveat as other variance reduced algorithms such as SDCA (Shalev-Shwartz & Zhang, 2013), SAG (Schmidt et al., 2013) or SAGA (Defazio et al., 2014). Despite the resemblance of our gradient estimator to the Stochastic Average Gradient (Schmidt et al., 2013), the convergence rate analyses are quite different.

3 Analysis

3.1 Preliminary tools

Recall that in our setting, our objective function is $\boldsymbol{w} \mapsto f(\boldsymbol{X}\boldsymbol{w})$, where $f(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^n f_i(\boldsymbol{\theta}_i)$. We suppose that for all i, f_i is L-smooth, which then implies that f satisfies the following non-standard smoothness condition:

$$\|\nabla f(\boldsymbol{\theta}) - \nabla f(\bar{\boldsymbol{\theta}})\|_p \leq \frac{L}{n}\|\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}\|_p \tag{6}$$

for every $p \in [1, \infty]$. Note that in this inequality – unlike in the standard definition of L-smoothness with respect to the ℓ_p norm – the same norm appears on both sides of the inequality. This inequality is proven in Appendix A. In particular it follows from (6) that f is (L/n)-smooth with respect to the ℓ_2 norm.

We therefore have the following quadratic upper bound on our objective function f, valid for all $\boldsymbol{w}, \boldsymbol{v}$ in the domain:

$$\begin{aligned} f(\boldsymbol{X}\boldsymbol{w}) \leq f(\boldsymbol{X}\boldsymbol{v}) + \langle \nabla f(\boldsymbol{X}\boldsymbol{v}), \boldsymbol{w} - \boldsymbol{v}\rangle \\ + \frac{L}{2n}\|\boldsymbol{X}\left(\boldsymbol{w} - \boldsymbol{v}\right)\|_2^2 . \end{aligned} \tag{7}$$

For $p \in \{1, 2, \infty\}$, we define the diameters

$$D_p = \max_{\boldsymbol{u}, \boldsymbol{v} \in \mathcal{C}} \|\boldsymbol{X}(\boldsymbol{u} - \boldsymbol{v})\|_p. \tag{8}$$

Remark 3. *For $p \in \{1, 2\}$, we have that $D_p^p \leq n D_\infty^p$.*

3.2 Worst-Case Convergence Rates for Smooth and Convex Objectives

We state our main result in the L-smooth, convex setting. In this section, we suppose that the f_is are L-smooth and convex and that for all $\boldsymbol{\theta}$, $f(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^n f_i(\boldsymbol{\theta}_i)$. The objective function f then satisfies (6) as noted previously.

Theorem 1. *Let $H_0 \overset{def}{=} \|\boldsymbol{\alpha}_0 - \nabla f(\boldsymbol{X}\boldsymbol{w}_0)\|_1$ be the initial error of our gradient estimator and $\boldsymbol{w}_\star \in \mathcal{C}$ a solution to OPT. We run Algorithm 1 with step sizes $\gamma_t = 2/(t+2)$. At time-step $t \geq 2$, the expected primal suboptimality $\mathbb{E}\varepsilon_t = \mathbb{E}[f(\boldsymbol{X}\boldsymbol{w}_t) - f(\boldsymbol{X}\boldsymbol{w}_\star)]$ has the following upper bound*

$$\begin{aligned} \mathbb{E}\varepsilon_t \leq & 2L\left(\frac{D_2^2 + 4(n-1)D_1 D_\infty}{n}\right)\frac{t}{(t+1)(t+2)} \\ & + \frac{2\varepsilon_0 + (2D_\infty H_0 + 64 L D_1 D_\infty)n^2}{(t+1)(t+2)} \end{aligned} \tag{9}$$

Remark 4. *The rate of the proposed method in terms of gradient calls is also given by (9) (one gradient call per iteration), whereas for deterministic Frank-Wolfe, the (deterministic) suboptimality after t gradient calls has the following upper bound (Jaggi, 2013; Hazan & Luo, 2016)*

$$\varepsilon_t \leq \frac{2LD_2^2}{t}. \tag{10}$$

In this paper, we will only discuss unit batch size. We can adapt our algorithm and proofs to consider sampling a mini-batch of b datapoints at each step. The leading term in our rate from Theorem 1 will change: we will use $\rho = 1 - \frac{b}{n}$ in Lemma 3. The overall rate will be modified accordingly. The per-iteration complexity will then become $\mathcal{O}(bd)$.

We first **sketch the outline of the proof** before delving into details. The proof of this convergence rate builds on three key lemmas. The first is an adaptation of Lemma 2 of Mokhtari et al. (2018) which bounds the suboptimality at step t by the sum of a contraction in the suboptimality at $t-1$, a vanishing term due to smoothness, and a last term depending on our gradient estimator's error in ℓ_1 norm. The first two terms show up in the convergence proof of the full-gradient Frank-Wolfe, see Lacoste-Julien & Jaggi (2015). The last term is an error, or noise term. Supposing the error term vanishes fast enough, we can fall back on the full-gradient proof technique (Frank & Wolfe, 1956; Jaggi, 2013).

From there, we show that the error term verifies a particular recursive inequality in lemma 2. In lemma 3, we then leverage this inequality to prove that the error term vanishes as $\mathcal{O}(1/t)$, finally allowing us to obtain the promised rate. The formal statements of these lemmas follow.

Lemma 1. *Let f_i be convex and L-smooth for all i. For* any *direction $\boldsymbol{\alpha}_t \in \mathbb{R}^n$, define $\boldsymbol{s}_t = \mathrm{LMO}(\boldsymbol{X}^\top \boldsymbol{\alpha}_t)$, $\boldsymbol{x}_t = (1-\gamma_t)\boldsymbol{x}_{t-1} + \gamma_t \boldsymbol{s}_t$ and $H_t = \|\boldsymbol{\alpha}_t - \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1})\|_1$.*

We have the following upper bound on the primal suboptimality at step t:

$$\varepsilon_t \leq (1-\gamma_t)\varepsilon_{t-1} + \gamma_t^2 \frac{LD_2^2}{2n} + \underbrace{\gamma_t D_\infty H_t}_{\textit{error term}}. \tag{11}$$

We defer this proof to Appendix B.

Remark 5. *This lemma holds for* any *direction $\boldsymbol{\alpha}_t \in \mathbb{R}^n$, not necessarily the $\boldsymbol{\alpha}_t$ given by the SFW algorithm.*

Remark 6. *This lemma generalizes the key inequality used in many proofs in the Frank-Wolfe literature (Jaggi, 2013) but includes an extra error term to account for the fact that the direction $\boldsymbol{\alpha}_t$, which we use for the* LMO *step and therefore to compute the updated iterate, is* not *the true gradient. If $\boldsymbol{\alpha}_t = \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1})$, that is, if we compute the gradient on the full dataset, then $H_t = 0$ and we recover the standard quadratic upper bound.*

In the following, $\boldsymbol{\alpha}_t$ is the direction given by Algorithm 1, and the ℓ_1 error term is in terms of that $\boldsymbol{\alpha}_t$:

$$H_t \stackrel{\text{def}}{=} \|\boldsymbol{\alpha}_t - \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1})\|_1 \tag{12}$$

for $t > 0$ and $H_0 = \|\boldsymbol{\alpha}_0 - \nabla f(\boldsymbol{X}\boldsymbol{w}_0)\|_1$.

Notice that we define the gradient estimator's error with the ℓ_1 norm. The previous lemma also holds with the ℓ_2 norm of the gradient error (replacing D_∞ by D_2). We prefer the ℓ_1 norm because of the finite-sum assumption: it induces a coordinate-wise separation over $\boldsymbol{\alpha}_t$ which corresponds to a datapoint-wise separation. The following lemma crucially leverages this assumption to upper bound H_t given by the SFW algorithm.

Lemma 2. *For the stochastic gradient estimator $\boldsymbol{\alpha}_t$ given by Algorithm 1 (SFW), we can upper bound $H_t = \|\boldsymbol{\alpha}_t - \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1})\|_1$ in conditional expectation as follows*

$$\mathbb{E}_t H_t \leq \left(1 - \frac{1}{n}\right)\left(H_{t-1} + \gamma_{t-1}\frac{LD_1}{n}\right). \tag{13}$$

Proof. We have the following expression for $\boldsymbol{\alpha}_t$, supposing that index i was sampled at step t.

$$\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1} + \left(\frac{1}{n} f_i'(\boldsymbol{x}_i^\top \boldsymbol{w}_{t-1}) - \boldsymbol{\alpha}_{t-1}^{(i)}\right)\boldsymbol{e}_i \tag{14}$$

where $\boldsymbol{e}_i$ is the i-th vector of the canonical basis of $\mathbb{R}^n$. Consider a fixed coordinate j. Since there is a $\frac{1}{n}$ chance of $\boldsymbol{\alpha}_j$ being updated to $f_j'(\boldsymbol{x}_j^\top \boldsymbol{w}_{t-1})$, taking conditional expectations we have

$$\mathbb{E}_t H_t^j \stackrel{\text{def}}{=} |\boldsymbol{\alpha}_t^{(j)} - \frac{1}{n} f_j'(\boldsymbol{x}_j^\top \boldsymbol{w}_{t-1})| \tag{15}$$

$$= \left(1 - \frac{1}{n}\right)|\boldsymbol{\alpha}_{t-1}^{(j)} - \frac{1}{n} f_j'(\boldsymbol{x}_j^\top \boldsymbol{w}_{t-1})|. \tag{16}$$

Summing over all coordinates we then have

$$\mathbb{E}_t H_t = \sum_{j=1}^n \mathbb{E}_t H_t^j \tag{17}$$

$$= \left(1 - \frac{1}{n}\right)\underbrace{\|\boldsymbol{\alpha}_{t-1} - \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1})\|_1}_{\delta_{t-1}}. \tag{18}$$

We denote the ℓ_1 norm term by δ_{t-1} for ease. Let us introduce the full gradient at the previous step $\nabla f(\boldsymbol{X}\boldsymbol{w}_{t-2})$ and use the triangle inequality. Our finite sum assumption gives us that for all $j \in \{1, \dots, n\}$ and $\boldsymbol{w} \in \mathcal{C}$, $[\nabla f(\boldsymbol{X}\boldsymbol{w})]_j = \frac{1}{n} f_j'(\boldsymbol{x}_j^\top \boldsymbol{w})$. Then, we separate the ℓ_1 norm, use L-smoothness of each of the f_js and the definition of $\boldsymbol{w}_{t-1}$.

$$\delta_{t-1} \leq H_{t-1} + \|\nabla f(\boldsymbol{X}\boldsymbol{w}_{t-2}) - \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1})\|_1 \tag{19}$$

$$\leq H_{t-1} + \frac{L}{n}\sum_{j=1}^n |\boldsymbol{x}_j^\top(\boldsymbol{w}_{t-1} - \boldsymbol{w}_{t-2})| \tag{20}$$

$$\leq H_{t-1} + \gamma_{t-1}\frac{L}{n}\sum_{j=1}^n |\boldsymbol{x}_j^\top(\boldsymbol{s}_{t-1} - \boldsymbol{w}_{t-2})| \tag{21}$$

$$\leq H_{t-1} + \gamma_{t-1}\frac{L}{n}\|\boldsymbol{X}(\boldsymbol{s}_{t-1} - \boldsymbol{w}_{t-2})\|_1 \tag{22}$$

where we used $\boldsymbol{w}_{t-1} - \boldsymbol{w}_{t-2} = \gamma_{t-1}(\boldsymbol{w}_{t-1} - \boldsymbol{s}_{t-2})$. Finally, using the definition of the diameter D_1, we obtain inequality (13). □

Now, we can use the structure of this recurrence to obtain the desired rate of convergence for our gradient estimator. We state this in the following lemma.

Lemma 3. *Let $\gamma_t = \frac{2}{t+2}$. We have the following bound on the expected error $\mathbb{E}H_t$, for $t \geq 2$:*

$$\mathbb{E}H_t \leq 2\frac{LD_1}{n}\left(\frac{2(n-1)}{t+2} + \left(1 - \frac{1}{n}\right)^{t/2}\log t\right) + \left(1 - \frac{1}{n}\right)^t H_0. \tag{23}$$

Remark 7. *Our gradient estimator's error in ℓ_1 norm goes to zero as $O\left(\frac{D_1}{t}\right)$. This rate depends on the assumption of the separability of f into a finite sum of L-smooth f_i's. On the other hand, it does not require that each (or any) f_i be convex.*

Proof. Consider a general sequence of nonnegative numbers, $u_0, u_1, u_2, \dots, u_t \in \mathbb{R}_+$ where for all t, the following recurrence holds:

$$u_t \leq \rho\left(u_{t-1} + \frac{K}{t+1}\right) \tag{24}$$

where $0 < \rho < 1$ and $K > 0$ are scalars.

First note that all the iterates are nonnegative. Suppose

$t \geq 2$,

$$\begin{aligned} u_t &\leq \rho^t u_0 + K \sum_{k=1}^{t} \frac{\rho^{t-k+1}}{k+1} \\ &= \rho^t u_0 + K \left(\sum_{k=1}^{\lfloor t/2 \rfloor} \frac{\rho^{t-k+1}}{k+1} + \sum_{k=\lfloor t/2 \rfloor + 1}^{t} \frac{\rho^{t-k+1}}{k+1} \right) \\ &\leq \rho^t u_0 + K \left(\sum_{k=1}^{\lfloor t/2 \rfloor} \frac{\rho^{t/2}}{k+1} + \sum_{k=\lfloor t/2 \rfloor + 1}^{t} 2 \, \frac{\rho^{t-k+1}}{t+2} \right). \end{aligned}$$

To go from the second line to the third line, we observe that for "old" terms with large steps sizes, we are saved by the higher power in the geometric term. For the more recent terms, the step-size is small enough to ensure convergence. More formally, in the early terms ($1 \leq k \leq \lfloor t/2 \rfloor$), we upper bound ρ^{t-k+1} by $\rho^{t/2}$. In the later terms ($\lfloor t/2 \rfloor + 1 \leq k \leq t$), we upper bound $\frac{1}{k+1}$ by $\frac{2}{t+2}$.

To obtain the full rate, we now study both parts separately. For the first part, we use knowledge of the harmonic series:

$$\rho^{t/2} \sum_{k=1}^{\lfloor t/2 \rfloor} \frac{1}{k+1} \leq \rho^{t/2} \log \left(\frac{t}{2} + 1 \right) \tag{25}$$

for $t \geq 2$, we can upper bound $\log \left(\frac{t}{2} + 1 \right)$ by $\log t$.

For the second part, we use knowledge of the geometric series:

$$\sum_{k=\lfloor t/2 \rfloor + 1}^{t} \rho^{t-k+1} \leq \frac{\rho}{1-\rho}. \tag{26}$$

Finally, for $t \geq 2$

$$0 \leq u_t \leq K \left(\frac{\rho}{(1-\rho)} \, \frac{2}{(t+2)} + \rho^{t/2} \log t \right) + \rho^t u_0. \tag{27}$$

The expected error $\mathbb{E}H_t$ verifies our general conditions with $u_0 = H_0 = \|\boldsymbol{\alpha}_0 - \nabla f(\boldsymbol{X}\boldsymbol{w}_{-1})\|_1$, defining $\boldsymbol{w}_{-1} \stackrel{\text{def}}{=} \boldsymbol{w}_0$ for the sake of the proof; $\rho = 1 - \frac{1}{n}$ and $K = \frac{2LD_1}{n}$. Specifying these values gives us the claimed bound. □

The remainder of the proof of Theorem 1 follows the usual Frank-Wolfe proofs in the full gradient case, which can be found e.g. in Frank & Wolfe (1956); Jaggi (2013). Here is a brief sketch of these steps: we tie the three key lemmas together, plugging in the bound on $\mathbb{E}H_t$ given by Lemma 3 into the upper bound on the suboptimality at step t given by Lemma 1. By specifying the step size $2/(t+2)$, and scaling the bounds by a factor of $(t+1)(t+2)$, we obtain a telescopic sum, allowing us to upper bound the expected suboptimality at the latest step considered. The details are deferred to Appendix C.

3.3 Worst-case Convergence Rates for Smooth, Non-Convex Objectives

We start by recalling the definition of the Frank-Wolfe gap:

$$g_t = \max_{\boldsymbol{s} \in \mathcal{C}} \langle \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1}), \boldsymbol{X}(\boldsymbol{w}_{t-1} - \boldsymbol{s}) \rangle. \tag{28}$$

Previous work (Jaggi, 2013) has shown the importance of the Frank-Wolfe gap. In the convex setting, it is a primal-dual gap, and as such, upper bounds both primal and dual suboptimalities. In the general non-convex setting, it is a measure of near-stationarity. We define a stationary point as any point $\boldsymbol{w}_\star$ such that for all $\boldsymbol{w} \in \mathcal{C}$, $\langle \nabla f(\boldsymbol{X}\boldsymbol{w}_\star), \boldsymbol{X}(\boldsymbol{w} - \boldsymbol{w}_\star) \rangle \geq 0$ (Bertsekas, 1999). From this definition, it is clear that the Frank-Wolfe gap g_t is zero only at a stationary point.

In this section, we suppose that f_i is L-smooth for i in $\{1, \ldots, n\}$, but not necessarily convex. The following theorem states that we can still obtain a stationary point from Algorithm 1.

Theorem 2. *Let $\boldsymbol{w}_t$ be computed according to Algorithm 1, then*

$$\liminf_{t \to \infty} \mathbb{E}_t g_t = 0, \tag{29}$$

where g_t is the Frank-Wolfe gap.

The proof of this result is deferred to Appendix F.

4 Stopping Criterion

In this section, we define a natural stochastic Frank-Wolfe gap, and explain why it can be used as a stopping criterion.

We recall the definition of the true Frank-Wolfe gap g_t, and define the stochastic Frank-Wolfe gap $\hat{g}_t$ as:

$$g_t = \max_{\boldsymbol{s} \in \mathcal{C}} \langle \nabla f(\boldsymbol{X}\boldsymbol{w}_{t-1}), \boldsymbol{X}(\boldsymbol{w}_{t-1} - \boldsymbol{s}) \rangle, \tag{30}$$

$$\hat{g}_t = \max_{\boldsymbol{s} \in \mathcal{C}} \langle \boldsymbol{\alpha}_t, \boldsymbol{X}(\boldsymbol{w}_{t-1} - \boldsymbol{s}) \rangle \tag{31}$$

for $\boldsymbol{\alpha}_t$ given by SFW.

The Frank-Wolfe gap's properties make estimating it very desirable: when the gap is small for a given iteration of a Frank-Wolfe type algorithm, we can guarantee we are close to optimum (or to a stationary point in the general non-convex case). Unfortunately, in datasets with many samples, and since it depends on the full gradient, computing this gap can be impractical.

The following proposition shows that the stochastic Frank-Wolfe gap estimator resulting from Algorithm 1 can be used as a proxy for the true Frank-Wolfe gap.

Proposition 1. *For $\boldsymbol{\alpha}_t$ given by Algorithm 1, we can bound the distance between the stochastic Frank-Wolfe gap and the true Frank-Wolfe gap as follows:*

$$|g_t - \hat{g}_t| \leq D_\infty H_t, \tag{32}$$

which yields the following bound in expectation

$$\mathbb{E}|g_t - \hat{g}_t| \leq 2\frac{LD_1D_\infty}{n}\left(\frac{2(n-1)}{t+2} + \left(1-\frac{1}{n}\right)^{t/2}\log t\right) + \left(1-\frac{1}{n}\right)^t D_\infty H_0. \tag{33}$$

We defer the proof to Appendix E.

If $\hat{g}_t$ goes to 0, then the true Frank-Wolfe gap will be expected to vanish as well. We therefore propose to use $\hat{g}_t$, which is computed as a byproduct of our SFW algorithm, as a heuristic stopping criterion, but defer a more in-depth theoretical and empirical analysis of this gap to future work.

5 Discussion

In this section, we compare the convergence rate of the proposed SFW, Lu & Freund (2018) and Mokhtari et al. (2018) as shown in Table 1. We use big $\mathcal{O}$ notation, only focusing on dependencies in n and t to upper bound the suboptimality at step t.

To make a fair comparison, including dependencies in n, the number of samples, we first standardize notations across papers. Lu & Freund (2018) use the same formal setting as ours, where $\boldsymbol{x}_i^\top \boldsymbol{w}$ is the argument to the i-th objective f_i, and the full objective is the average of these. Mokhtari et al. (2018) set themselves in a more general setting, where they only assume access to an *unbiased estimator* of the full gradient.

For ease of comparison, we rewrite the two algorithms of Lu & Freund (2018) and Mokhtari et al. (2018) in Appendix G using our notations.

Because of their more general setting, the L_{mok} Lipschitz constant appearing in Mokhtari et al. (2018) can be written $L_{\text{mok}} = \frac{L}{n} n \max_i \|\boldsymbol{x}_i\|_2$ (using Cauchy-Schwartz). Their diameter constant $D_{\text{mok}} = \max_{\boldsymbol{u},\boldsymbol{v}\in\mathcal{C}} \|\boldsymbol{u}-\boldsymbol{v}\|_2$ is also independent of n. Finally, their σ^2 term controlling the variance of their stochastic estimator should also be n-independent. Under this notation, their convergence rate (Theorem 3, Mokhtari et al. (2018)) is $\mathcal{O}\left(1/\sqrt[3]{t}\right)$ with no dependency in n as expected.

Lu & Freund (2018) have a detailed discussion of the rate of their method, and achieve the overall rate of $\mathcal{O}\left(n/t\right)$.

To fairly compare these rates to the one given by Theorem 1, we must consider the D_1 and D_∞ terms, which may depend on the number of samples n. The rate we obtain has a leading term of $\mathcal{O}\left(D_1D_\infty/t\right)$, and a second term of $\mathcal{O}\left(D_1D_\infty n^2/t^2\right)$. The second term is dominated by the first in the regime $t > n^2$. Defining $\kappa = D_1/D_\infty$, we can write D_1D_∞ as κD_∞^2. We have that $\kappa \leq n$, meaning that in the worst case, this bound matches the one in Lu & Freund (2018). When the constraint set is the ℓ_1 ball $\{\boldsymbol{w} \mid \|\boldsymbol{w}\|_1 \leq \lambda\}$, we have the following closed form expression:

$$\kappa = \frac{\|X\|_{1,1}}{\|X\|_{1,\infty}} = \frac{\max_j \sum_{i=1}^n |X_{ij}|}{\max_{ij} |X_{ij}|}. \tag{34}$$

We can therefore easily compute it for given datasets.

Remark 8. *We briefly remark that if for every feature, the contribution of that feature is limited to a few datapoints, this ratio will be small, and therefore the overall bound does not depend on the number of samples. This tends to happen for TF-IDF text representations, and for fat-tailed data.*

Formal analysis of this ratio exceeds the scope of this paper, and we defer it to future work. We report values of κ for the considered datasets in Section 7.

6 Implementation Details

Our implementation is available in the C-OPT package.[1]

Initialization. We use the cheapest possible initialization: our initial stochastic gradient estimator $\boldsymbol{\alpha}_0$ starts out at 0. We also then have that $\boldsymbol{r}_0 = 0$.

Sparsity in $\boldsymbol{X}$. Suppose there are at most s non-zero features for any datapoint $\boldsymbol{x}_i$. Then for instances where $\mathcal{C}$ is an ℓ_1 ball, all updates in SFW algorithm can be implemented using using only the support of the current datapoint, making the per-iteration cost of SFW $\mathcal{O}(s)$ instead of $\mathcal{O}(d)$. Large-scale datasets are often extremely sparse, so leveraging this sparsity is crucial. For example, in the LibSVM datasets suite, 8 out of the 11 datasets with more than a million samples have a density between 10^{-4} and 10^{-6}.

7 Experiments

We compare the proposed SFW algorithm with other constant batch size algorithms from Mokhtari et al. (2018) and Lu & Freund (2018).

Experimental Setting. We consider ℓ_1 constrained logistic regression problems on the BREAST CANCER and RCV1 datasets, and an ℓ_1 constrained least squares regression problem on the CALIFORNIA HOUSING dataset, all from the

[1] https://github.com/openopt/copt

Table 2. Datasets and tasks used in experiments.

DATASET	n	d	κ/n	f_i	$\mathcal{C}$
BREAST CANCER	683	10	0.929	$\log(1+\exp(-\boldsymbol{y}_i \boldsymbol{x}_i^\top \boldsymbol{w}))$	$\{\|\boldsymbol{w}\|_1 \leq \lambda,\ \lambda = 5\}$
RCV1	20,242	47,236	0.021	$\log(1+\exp(-\boldsymbol{y}_i \boldsymbol{x}_i^\top \boldsymbol{w}))$	$\{\|\boldsymbol{w}\|_1 \leq \lambda,\ \lambda = 100\}$
CALIFORNIA HOUSING	20,640	8	0.040	$\frac{1}{2}(\boldsymbol{y}_i - \boldsymbol{x}_i^\top \boldsymbol{w})^2$	$\{\|\boldsymbol{w}\|_1 \leq \lambda, \lambda = 0.1\}$

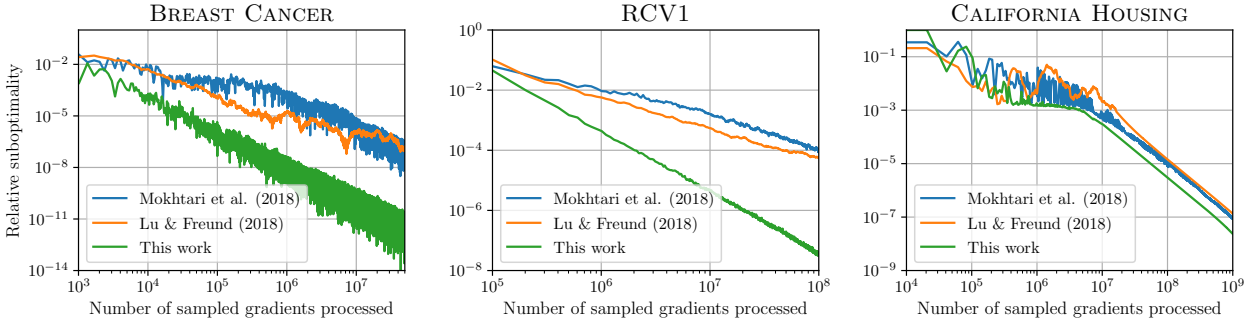

Figure 1. Comparing our SFW method to the related works of Lu & Freund (2018) and Mokhtari et al. (2018). From left to right: BREAST CANCER, RCV1, and CALIFORNIA HOUSING datasets. We plot the relative subtimality values in log-log plots to show empirical rates of convergence. We use the following batch size: $b = \lfloor n/100 \rfloor$.

UCI dataset repository (Dua & Graff, 2017). See Table 2 for details and links.

We compare the relative suboptimality computed for each method, given by $(f(\boldsymbol{X}\boldsymbol{w}_t) - f_{\min})/(f_{\max} - f_{\min})$ at step t, where $f_{\min}$ and $f_{\max}$ are the smallest and largest function values encountered by any of the compared methods. We compute these values at different time intervals (the same for each method) depending on problem size, to limit the time of each run. We use batches using 1% of the dataset at each step, following Lu & Freund (2018). Within a batch, data points are sampled without replacement.

We plot these values as a function of the number of gradient evaluations, equal to the number of iterations times the batch size b: for all of the considered methods, an iteration involves exactly b gradient evaluations and one call to the LMO. This allows us to fairly compare the convergence speeds in practice.

Compared to both methods from Mokhtari et al. (2018) and Lu & Freund (2018), the proposed SFW achieves lower suboptimality for a given number of iterations on the considered tasks and datasets. We have no explanation for the initial regime in the CALIFORNIA HOUSING dataset, before the methods start showing what resembles a sublinear rate, as the theory prescribes. Notice that the RCV1 dataset has the lowest κ/n (due to sparsity of the TF-IDF represented data), and that the method presented in this paper performs particularly well on this dataset.

Comparison with Mokhtari et al. (2018). Although the step-size in our SFW Algorithm and the one proposed in the paper are of the same order of magnitude $\mathcal{O}(1/t)$, Mokhtari et al. (2018) use $f_i'(\boldsymbol{x}_i^\top \boldsymbol{w}_{t-1})$ instead of our $(1/n) f_i'(\boldsymbol{x}_i^\top \boldsymbol{w}_{t-1})$, because they require an unbiased estimator. Their choice induces higher variance, which then requires the algorithm to use momentum with a vanishing step size in their stochastic gradient estimator, damping the contributions of the later gradients (using $\rho_t = \frac{1}{t^{2/3}}$, see the pseudo code in Appendix G). This may explain why the method proposed in Mokhtari et al. (2018) achieves slower convergence. On the contrary, the lower variance in our estimator $\boldsymbol{\alpha}_t$ allows us to give the same weight to contributions of later gradients as to previous ones, and to forget all but the last gradient computed at a given datapoint.

Comparison with Lu & Freund (2018). The method from Lu & Freund (2018) computes the gradient at an averaged iterate, putting more weight on earlier iterates, making it more conservative. This may explain slower convergence versus the SFW algorithm proposed in this paper in certain settings.

8 Conclusion and Future Work

Similarly to methods from the Variance Reduction literature such as SAG, SAGA, SDCA, we propose a Stochastic Frank Wolfe algorithm tailored to the finite-sum setting. Our method achieves a step towards attaining comparable com-

plexity iteration-wise to deterministic, true-gradient Frank-Wolfe in the smooth, convex setting, at a per-iteration cost which can be nearly independent of the number of samples in the dataset in favorable settings. Our rate of convergence depends on the norm ratio κ on the dataset, which is related to a measure of the weights of the data distribution's tails. We will explore this intriguing fact in future work.

We propose a stochastic Frank-Wolfe gap estimator, which may be used as a heuristic stopping criterion, including in the non-convex setting. Its distance to the true gap may be difficult to evaluate numerically. Obtaining a practical bound on this distance is an interesting avenue for future work.

Guélat & Marcotte (1986) and Lacoste-Julien & Jaggi (2015) have proposed variants of the FW algorithm that converge linearly on polytope constraint sets for strongly convex objectives: the Away Steps Frank-Wolfe and the Pairwise Frank-Wolfe. Goldfarb et al. (2017) studied stochastic versions of these and showed linear convergence over polytopes using increasing batch sizes. Our SFW algorithm, the natural stochastic gap and the analyses in this paper should be amenable to such variants as well, which we plan to explore in future work.

Acknowledgments

The authors would like to thank Donald Goldfarb for early encouragement in this direction of research, and Armin Askari, Sara Fridovich-Keil, Yana Hasson, Thomas Kerdreux, Nicolas Le Roux, Romain Lopez, Grégoire Mialon, Courtney Paquette, Hector Roux de Bézieux, Alice Schoenauer-Sebag, Dhruv Sharma, Yi Sun, and Nilesh Tripuraneni for their constructive criticism on drafts of this paper. The authors also warmly thank Maria-Luiza Vladareanu for finding and reporting an error in an earlier draft's proof, and Alex Belloni, Jose Moran for discussions as well.

Francesco Locatello is supported by the Max Planck ETH Center for Learning Systems, by an ETH core grant (to Gunnar Rätsch), and by a Google Ph.D. Fellowship. Robert Freund's research is supported by AFOSR Grant No. FA9550-19-1-0240.

References

Abernethy, J., Lai, K. A., Levy, K. Y., and Wang, J.-K. Faster Rates for Convex-Concave Games. In *Proceedings of the 31st Conference On Learning Theory*, 2018.

Abernethy, J. D. and Wang, J.-K. On Frank-Wolfe and Equilibrium Computation. In *Advances in Neural Information Processing Systems 30*, 2017.

Bertsekas, D. P. *Nonlinear programming*. Athena Scientific, 1999.

Defazio, A., Bach, F., and Lacoste-Julien, S. SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives. In *Advances in Neural Information Processing Systems 27*. 2014.

Demyanov, V. and Rubinov, A. The minimization of a smooth convex functional on a convex set. *SIAM Journal on Control*, 1967.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Frank, M. and Wolfe, P. An algorithm for quadratic programming. *Naval Research Logistics (NRL)*, 1956.

Goldfarb, D., Iyengar, G., and Zhou, C. Linear Convergence of Stochastic Frank Wolfe Variants. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

Guélat, J. and Marcotte, P. Some comments on Wolfe's 'away step'. *Mathematical Programming*, 1986.

Hazan, E. and Luo, H. Variance-reduced and projection-free stochastic optimization. In *International Conference on Machine Learning*, 2016.

Hofmann, T., Lucchi, A., Lacoste-Julien, S., and McWilliams, B. Variance reduced stochastic gradient descent with neighbors. In *Advances in Neural Information Processing Systems*, 2015.

Jaggi, M. Revisiting Frank-Wolfe: projection-free sparse convex optimization. In *International Conference on Machine Learning*, 2013.

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

Levitin, E. S. and Polyak, B. T. Constrained minimization methods. *USSR Computational mathematics and mathematical physics*, 1966.

Locatello, F., Khanna, R., Tschannen, M., and Jaggi, M. A Unified Optimization View on Generalized Matching Pursuit and Frank-Wolfe. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

Locatello, F., Yurtsever, A., Fercoq, O., and Cevher, V. Stochastic Frank-Wolfe for Composite Convex Minimization. In *Advances in Neural Information Processing Systems 32*. 2019.

Lu, H. and Freund, R. M. Generalized Stochastic Frank-Wolfe Algorithm with Stochastic "Substitute" Gradient for Structured Convex Optimization. *arXiv*, 2018.

Mokhtari, A., Hassani, H., and Karbasi, A. Stochastic Conditional Gradient Methods: From Convex Minimization to Submodular Maximization. *arXiv*, 2018.

Reddi, S. J., Sra, S., Póczos, B., and Smola, A. Stochastic Frank-Wolfe methods for nonconvex optimization. In *54th Annual Allerton Conference on Communication, Control, and Computing*, 2016.

Schmidt, M., Le Roux, N., and Bach, F. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 2013.

Shalev-Shwartz, S. and Zhang, T. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 2013.

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

Zhang, M., Shen, Z., Mokhtari, A., Hassani, H., and Karbasi, A. One Sample Stochastic Frank-Wolfe. *arXiv*, 2019.