USING TAYLOR-APPROXIMATED GRADIENTS TO IMPROVE THE FRANK-WOLFE METHOD FOR EMPIRICAL RISK MINIMIZATION

ZIKAI XIONG* AND ROBERT M. FREUND†

Abstract. The Frank-Wolfe method has become increasingly useful in statistical and machine learning applications, due to the structure-inducing properties of the iterates, and especially in settings where linear minimization over the feasible set is more computationally efficient than projection. In the setting of Empirical Risk Minimization – one of the fundamental optimization problems in statistical and machine learning – the computational effectiveness of Frank-Wolfe methods typically grows linearly in the number of data observations n. This is in stark contrast to the case for typical stochastic projection methods. In order to reduce this dependence on n, we look to second-order smoothness of typical smooth loss functions (least squares loss and logistic loss, for example) and we propose amending the Frank-Wolfe method with Taylor series-approximated gradients, including variants for both deterministic and stochastic settings. Compared with current state-of-the-art methods in the regime where the optimality tolerance ε is sufficiently small, our methods are able to simultaneously reduce the dependence on large n while obtaining optimal convergence rates of Frank-Wolfe methods, in both the convex and non-convex settings. We also propose a novel adaptive step-size approach for which we have computational guarantees. Last of all, we present computational experiments which show that our methods exhibit very significant speed-ups over existing methods on real-world datasets for both convex and non-convex binary classification problems.

Key words. Frank-Wolfe, linear minimization oracle, Empirical Risk Minimization, linear prediction, computational complexity, convex optimization

AMS subject classifications. 90C25, 90C26, 90C06, 90C60, 68Q25

1. Introduction.

Our problem of interest is the following Empirical Risk Minimization (ERM) problem (also called the *finite-sum* problem) with constraints:

$$\text{ERM:} \qquad \text{minimize}_{x\in\mathcal{C}} \quad F(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x) \ , \tag{1.1}$$

where $f_i(\cdot) : \mathbb{R}^p \to \mathbb{R}$ is the loss function of observation i for $i = 1, \ldots, n$, and x is comprised of the (model) coefficients/parameters which are to be optimized over the feasibility set $\mathcal{C} \subset \mathbb{R}^p$. Problem (1.1) is a fundamental problem in statistical and machine learning, as it is the underlying optimization problem on which models are "trained" or "learned", see [27, 10].

We are especially interested in and will focus in this study on those empirical risk minimization problems where the loss functions exhibit "linear prediction" [18], namely each loss function is of the form $f_i(x) := l_i(w_i^\top x)$ where w_i is the i-th feature vector and $l_i(\cdot) : \mathbb{R} \to \mathbb{R}$ is a univariate function; thus f_i is a scalar function of the linear function $w_i^\top x$. In this case (1.1) becomes

$$\text{ERM}_\ell : \quad \text{minimize}_{x\in\mathcal{C}} \quad F(x) := \frac{1}{n}\sum_{i=1}^{n} \left[f_i(x) = l_i(w_i^\top x)\right] \ , \tag{1.2}$$

and we refer to (1.2) as "ERM$_\ell$" for "ERM with linear prediction", because the i-th predicted value of interest is the linear function $w_i^\top x$ of the model coefficients x. This

*MIT Operations Research Center, 77 Massachusetts Avenue, Cambridge, MA 02139, USA (zikai@mit.edu). Research supported by AFOSR Grant No. FA9550-19-1-0240.

†MIT Sloan School of Management, 77 Massachusetts Avenue, Cambridge, MA 02139, USA (rfreund@mit.edu). Research supported by AFOSR Grant No. FA9550-19-1-0240.

structure arises in many canonical applications of statistical and machine learning problems, including support vector machines (SVMs), LASSO, logistic regression, and matrix completion, among others, see [20, 18] for other examples. Let $W \in \mathbb{R}^{p\times n}$ be the matrix whose columns are comprised of the n feature vectors $w_1,\ w_2,\ \ldots,\ w_n$ in (1.2). We also define the function $l(\cdot) : \mathbb{R}^n \to \mathbb{R}$ as $l(\theta) := \frac{1}{n}\sum_{i=1}^n l_i(\theta_i)$, and so the objective function of ERM_ℓ can be written as $F(x) = l(W^\top x)$ with $(\theta_1, \ldots, \theta_n) = \theta = W^\top x$. Let $w_i(\mathcal{C}) \subset \mathbb{R}$ denote the range of $w_i^\top x$ over $x \in \mathcal{C}$. We make the following assumptions for ERM_ℓ throughout this paper:

Assumption 1.1. The following conditions hold for the ERM_ℓ problem (1.2):

(i) the feasibility set $\mathcal{C}$ is a compact convex set,

(ii) for any $g \in \mathbb{R}^p$, the linear minimization problem $\arg\min_{x\in\mathcal{C}} g^\top x$ can be easily solved,

(iii) for all $i \in [n]$ the univariate loss function $l_i(\cdot)$ is twice-differentiable and its first-order derivative $l_i'(\cdot)$ is L-Lipschitz continuous on the range of $w_i^\top x$ over $x \in \mathcal{C}$, namely

$$|l_i'(\bar\theta) - l_i'(\hat\theta)| \le L|\bar\theta - \hat\theta| \quad \text{for any } \bar\theta, \hat\theta \in w_i(\mathcal{C}), \text{ and} \tag{1.3}$$

(iv) for all $i \in [n]$ the second-order derivative $l_i''(\cdot)$ is $\hat L$-Lipschitz continuous on the range of $w_i^\top x$ over $x \in \mathcal{C}$, namely

$$|l_i''(\bar\theta) - l_i''(\hat\theta)| \le \hat L|\bar\theta - \hat\theta| \quad \text{for any } \bar\theta, \hat\theta \in w_i(\mathcal{C})\ . \tag{1.4}$$

The first three conditions of Assumption 1.1 are canonical for Frank-Wolfe and other methods that rely on a Linear Minimization Oracle (LMO). However, condition (iv) of Assumption 1.1 concerns the smoothness of the second derivatives of the loss functions, and does not typically appear in the literature. In Section 1.1 we discuss and give some justification for this extra condition.

1.1. Examples of oft-used loss functions with smooth second derivatives. Here we discuss three oft-used loss functions that have smooth second derivatives and therefore satisfy condition (iv) of Assumption 1.1.

Quadratic losses. In very many practical instances of ERM_ℓ the loss functions are of the form $l_i(\theta_i) = \frac{1}{2}(y_i - \theta_i)^2$ for some observed value y_i, such as in LASSO instances [26], matrix completion with quadratic losses, and structured sparse matrix estimation with CUR factorization [19, 18]. We note that $L = 1$ and $\hat L = 0$ for quadratic losses.

Logistic regression for binary classification. In logistic regression we are given a training set $\{(w_i, y_i)\}_{i=1}^n$ with $w_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$, and we solve the maximum likelihood estimation problem by computing the optimal solution to the following logistic regression optimization problem:

$$\min_{x\in\mathcal{C}}\ F(x) := \tfrac{1}{n}\sum_{i=1}^n \ln(1 + \exp(-y_i w_i^\top x))\ . \tag{1.5}$$

In this problem, $F(x) := \frac{1}{n}\sum_{i=1}^n l_i(w_i^\top x)$ and $l_i(\theta_i) := \ln(1 + \exp(-y_i\theta_i)$. For each $i \in [n]$ and simplifying notation using $v = \theta_i$, we have $l_i(v) = \ln(1 + \exp(-y_i v))$, $l_i'(v) = y_i\frac{-1}{1+\exp(y_i v)}$, $l_i''(v) = y_i^2\frac{\exp(y_i v)}{(1+\exp(y_i v))^2}$, and $l_i'''(v) = y_i^3\frac{\exp(y_i v)-\exp(2y_i v)}{(1+\exp(y_i v))^3}$. We note that $|l_i''(v)|$ obtains its maximal value of $\frac{1}{4}$ at $v = 0$, and hence $L = \frac{1}{4}$. Similarly, $|l_i'''(v)|$ obtains its maximal value of $\frac{1}{6\sqrt{3}}$ when $\exp(y_i v) = 2 - \sqrt{3}$ and so $\hat L = \frac{1}{6\sqrt{3}}$. In summary, l_i' is $\frac{1}{4}$-Lipschitz continuous and l_i'' is $\frac{1}{6\sqrt{3}}$-Lipschitz continuous for $i \in [n]$.

Binary linear classification with non-convex losses [20]. We are given a training set $\{(w_i, y_i)\}_{i=1}^n$ with $w_i \in \mathbb{R}^p$ and $y_i \in \{0,1\}$, and the task is to learn $\mathbb{P}[Y = 1|w = w_i] = \sigma(w_i^\top x)$ for a given threshold function $\sigma : \mathbb{R} \to [0,1]$. This leads to the optimization problem:

$$\min_{x\in\mathcal{C}} \ \frac{1}{n}\sum_{i=1}^{n}(y_i - \sigma(w_i^\top x))^2 \ , \tag{1.6}$$

where $F(x) := \frac{1}{n}\sum_{i=1}^n l_i(w_i^\top x)$, $l_i(\cdot) := (y_i - \sigma(\cdot))^2$, and $\sigma : \mathbb{R} \to [0,1]$ is the given threshold function, such as the sigmoid function $\sigma(v) = (1+\exp(-v))^{-1}$. We note in general that (1.6) is a non-convex optimization problem. We have $l_i'(v) = -2\sigma'(v)(y_i - \sigma(v))$, $l_i''(v) = 2(\sigma'(v))^2 - 2\sigma''(v)(y_i - \sigma(v))$, and $l_i'''(v) = 6\sigma'(v)\sigma''(v) - 2\sigma'''(v)(y_i - \sigma(v))$. Suppose that there exists a constant L_σ such that $|\sigma'(v)|$, $|\sigma''(v)|$, and $|\sigma'''(v)| \le L_\sigma$ for any $v \in \mathbb{R}$. Then it follows that l_i' is $L := (2L_\sigma^2 + 2L_\sigma)$-Lipschitz continuous and l_i'' is $\hat{L} := (6L_\sigma^2 + 2L_\sigma)$-Lipschitz continuous.

In the often-encountered case when σ is the sigmoid function, we have $\sigma'(x) = \frac{\exp(x)}{(1+\exp(x))^2}$, $\sigma''(x) = \frac{\exp(x)(\exp(x)-1)}{(1+\exp(x))^3}$, and $\sigma'''(x) = \frac{\exp(x)(1-4\exp(x)+\exp(2x))}{(1+\exp(x))^4}$, and thus $|\sigma'(x)| \le 1/4$, $|\sigma''(x)| \le 1/6\sqrt{3}$ and $|\sigma'''(x)| \le 1/24$. Because $y_i \in \{0,1\}$ and $\sigma(x) \in [0,1]$, $|y_i - \sigma(x)| \le 1$, it holds that $|l_i''(x)| \le \frac{1}{8} + \frac{1}{3\sqrt{3}}$ and $|l_i'''(x)| \le \frac{1}{4\sqrt{3}} + \frac{1}{12}$. Therefore l_i' is $(\frac{1}{8} + \frac{1}{3\sqrt{3}})$-Lipschitz continuous and l_i'' is $(\frac{1}{4\sqrt{3}} + \frac{1}{12})$-Lipschitz continuous.

1.2. Motivation and literature review. With an ever-increasing number n of data observations in statistical and machine learning applications in ERM and ERM_ℓ, there is growing importance in solving models where the number of observations n is very large (compared, say, to the dimension p of the features. The presence of large n does not pose a particular problem (at least in theory) for a typical version of the stochastic gradient descent method (SGD), which forms a gradient estimate at each iteration by randomly sampling a mini-batch of observations and then projects back onto the feasibility set $\mathcal{C}$ after taking a gradient-estimate-based step. The computational complexity of SGD is not adversely affected by large values of n.

However, in many applications of ERM and ERM_ℓ it is more attractive to use the Frank-Wolfe method or a version thereof, due to the structure-inducing properties of the iterates (sparsity, low-rank) especially when combined with in-face steps, see for example [7]. The Frank-Wolfe method is also attractive in settings where a Linear Minimization Oracle (LMO) on the feasibility set $\mathcal{C}$ is more computationally efficient than projection onto $\mathcal{C}$, where an LMO is a subroutine that solves and returns

$$\arg\min_{x\in\mathcal{C}} \langle g, x\rangle$$

for any given objective function vector $g \in \mathbb{R}^p$. For example, when $\mathcal{C}$ is the nuclear norm ball, the LMO only requires computing the leading singular vector of a matrix, as compared to a projection oracle which requires a full SVD of a matrix, see [13] for example. As a result, the Frank-Wolfe method, which replaces gradient steps and projection by an LMO instead, has become increasingly useful in statistical and machine learning applications, for both convex and non-convex loss functions. However, when the number of observations n is large, the Frank-Wolfe method has lacked "good" efficiency as compared to SGD, as the number of operations of Frank-Wolfe typically grows as $O(n/\varepsilon)$. The primary motivation of the research in this paper therefore is to address the following question:

Are there efficient stochastic (or deterministic) Frank-Wolfe methods

that eliminate or reduce the dependence on the number of observations n, in theory and in practice?

In the last decade many researchers have studied stochastic versions of the Frank-Wolfe method. Before discussing these related works, let us introduce some useful measurements for Frank-Wolfe methods. We use the following Frank-Wolfe gap $\mathcal{G}(x)$ as a measure of non-stationarity:

$$\mathcal{G}(x) := \max_{s \in \mathcal{C}} \langle x - s, \nabla F(x) \rangle \ . \tag{1.7}$$

The Frank-Wolfe gap is always nonnegative and is 0 if and only if x is a stationary point of (1.2). For problems with convex losses F, supposing that $x^\star$ is an optimal solution, we say x is an ε-optimal solution if and only if the optimality gap $F(x) - F(x^\star)$ is no larger than ε. Since the Frank-Wolfe gap is an upper bound on the optimality gap when F is convex, $\mathcal{G}(x^k)$ is a conservative measure of the optimality gap at x^k, see [13] for example. The Frank-Wolfe gap does not necessarily bound the optimality gap in the non-convex setting; nevertheless $\mathcal{G}(x)$ is always nonnegative and $\mathcal{G}(x) = 0$ if and only if x is a stationary point of (1.2), and for this reason $\mathcal{G}(x)$ is often used as a measure of non-stationarity at x, see [15, 23, 29, 22]. For an instance of (1.2) with non-convex objective function F, we therefore say that a point $x \in \mathcal{C}$ is an ε-stationarity point of (1.2) if $\mathcal{G}(x) \leq \varepsilon$.

Generally speaking, for deterministic versions of the Frank-Wolfe method the lower bound LMO complexity to obtain an ε-optimal solution is $O(1/\varepsilon)$, see [13, 16]. And in the case when F is non-convex, the LMO complexity is at least $O(1/\varepsilon^2)$, see [25]. These deterministic methods require access to exact full-batch gradients per iteration, whose complexity scales with the number of observations n. When n is very large, computing exact gradients might be no longer efficient. We refer the reader to [6, 13, 17, 15] among other recent deterministic versions of the Frank-Wolfe method in this context.

Stochastic Frank-Wolfe methods avoid computing the exact gradient in each iteration by doing stochastic gradient estimation aimed at reducing the dependence on n in the overall computational complexity. Most existing stochastic Frank-Wolfe methods typically replace exact gradients by the average of sampled gradients as recorded in the current or previous iterates, so that they can reduce (or eliminate) the dependence on n in the overall complexity. Unfortunately the approaches that eliminate the dependence on n do so at the expense of a higher overall LMO complexity which tends to grow much faster than $O(1/\varepsilon)$; see [11, 17, 21] for representative work in this vein. A second approach is to estimate gradients by the variance reduction techniques which emerged from other stochastic first-order methods, such as SVRG [14] and SPIDER [4]. These methods still periodically require access to the exact full gradient, and they also compute averages from a gradually increasing number of gradient samples, see [11, 23, 29, 25, 9]. Also unfortunately, some of these methods perform worse in practice than the methods with even weaker theoretical guarantees [11]. A third approach is based on the primal-dual structure of ERM_ℓ when f is convex. However, depending on the specific assumptions and problem set-up, the overall dependence of these methods on ε and n is still $O(n/\varepsilon)$, which is the same as the traditional (deterministic) Frank-Wolfe method, see [18, 22]. In addition to the above approaches, some methods avoid directly computing gradients by estimating them via zeroth-order information (see, e.g., [8, 12, 1, 24]). The theoretical complexities of these methods are usually no better than the methods using gradients. Recently, a new line of research has emerged that seeks to estimate gradients with the aid of second-order informa-

tion, see for example [25, 30, 9, 30]. However, the cost per iteration in these methods may still grow fast, or they still periodically require access to the exact gradient. In summary, there are no existing stochastic Frank-Wolfe methods (that we are aware of) whose overall complexity does not scale with n while maintaining a dependence on ε of only $O(1/\varepsilon)$ (for convex losses) or $O(1/\varepsilon^2)$ (for non-convex losses).

1.3. Contributions.

- In the setting of ERM_ℓ we develop a new family of Frank-Wolfe methods, which we call TUFW for "Taylor-point Updating Frank-Wolfe," that replaces the exact gradient computation by a sum of (second-order) Taylor-approximated gradients around some current and previous iterates, which we call the *Taylor points*.
- Different versions of TUFW are developed based on different rules for constructing the batches of observations for Taylor-point updating at each iteration. For both convex and non-convex losses, we propose both stochastic and deterministic rules. Our rules exhibit a *decreasing* number of gradient calls over the course of the intended iterations, while retaining the optimal LMO complexity of $O(1/\varepsilon)$ (for convex losses) and $O(1/\varepsilon^2)$ (for non-convex losses), together with other flops ($O(n/\sqrt{\varepsilon})$ for convex losses and $O(n/\varepsilon^{3/2})$ for non-convex losses. In the regime when ε is sufficiently small these other flops are minor compared with the LMO complexity. Our stochastic TUFW also avoids periodically computing exact gradients.
- We present computational experiments which show that our methods exhibit very significant speed-ups over existing methods on real-world datasets for convex and non-convex binary classification problems.
- Our TUFW method easily extends to the more general ERM problem, with corresponding versions of our theoretical guarantees.
- We also propose a novel adaptive step-size method which has similar theoretical guarantees, and for which our computational experiments indicate superior performance in practice.

1.4. Outline. In section 2 we present our new type of Frank-Wolfe method, which we call TUFW for "Taylor-point Updating Frank-Wolfe", along with some elementary properties of the method. In section 3 we present computational guarantees and overall complexity of TUFW for convex losses, and in section 4 we present computational guarantees and overall complexity for non-convex losses. section 6 contains computational experiments for both convex and non-convex instances of ERM_ℓ.

1.5. Notation. Let $\mathbb{N}$ denote the natural numbers. For any $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. For any real number v, we use $\lfloor v \rfloor$ to denote the largest integer number less than or equal to v. Depending on the context, we will use either $\langle a, b\rangle$ or $a^\top b$ to denote the inner product between a and b in Euclidean space. For a vector x in $\mathbb{R}^p$, the norm $\|x\|_q$ is given by $(\sum_{j=1}^p x_j^q)^{1/q}$ for $q \in [1, \infty)$ and $\|x\|_\infty := \max_j |x_j|$. The dual of norm $\|\cdot\|$ is defined as $\|z\|_* := \sup\{z^\top x : \|x\| \le 1\}$. For a matrix $A \in \mathbb{R}^{p\times p}$, the matrix operator norm $\|A\|$ induced by the norm $\|\cdot\|$ is defined as $\sup_{x:\|x\|\le 1} \|Ax\|_*$. For any symmetric positive definite matrix $A \in \mathbb{S}^{p\times p}$, we define its induced norm on $\mathbb{R}^p$ by $\|z\|_A := \sqrt{z^\top Az}$. We define $\sum_{i=j}^k \alpha_i := 0$ whenever $j > k$. We denote the uniform distribution on the set S by $\mathcal{U}(S)$, and we denote the Bernoulli distribution with probability p by $\text{Ber}(p)$.

2. A Frank-Wolfe method with Taylor-point Updating (TUFW). In this section we present a new type of Frank-Wolfe algorithm whose newness derives

from the way in which approximate gradients are computed. We call the algorithm TUFW for "Taylor-point Updating Frank-Wolfe". Algorithm TUFW is similar to the standard Frank-Wolfe method except that it computes an inexpensive estimate g^k of the gradient $\nabla F(x^k)$ at iteration k. And it differs from other Frank-Wolfe methods in the way it computes the estimate g^k in that (i) TUFW explicitly uses second-derivatives of the loss functions l_i, and (ii) it can use very different batch-size rules – both stochastic and deterministic. TUFW is described in Algorithm 2.1. We now explain the steps of the method.

Algorithm 2.1 (TUFW) Frank-Wolfe Method with Taylor Point Updating

1: **input** initial point x^0, step-size sequence $\{\gamma_k\}_k$, and (possibly) number of iterations K
2: **initialize Taylor points** $\mathcal{B}_0 \leftarrow [n]$ and $b_i \leftarrow x^0$ for $i \in [n]$
3: **for** $k = 0, 1, 2, \ldots, K$, **do**
4: **If** $k > 0$, **update Taylor points**:
Use some **Rule** to construct set $\mathcal{B}_k \subset [n]$ of indices for Taylor-point updating
Update Taylor points: for $i \in \mathcal{B}_k$ update $b_i \leftarrow x^k$
5: **Compute estimates of individual gradients**:
$g_i^k \leftarrow \nabla f_i(b_i) + \nabla^2 f_i(b_i)(x^k - b_i)$ for $i \in [n]$
6: **Compute estimate of full gradient**: $g^k \leftarrow \frac{1}{n}\sum_{i=1}^n g_i^k$
7: **Solve LMO**: $s^k \leftarrow \arg\min_{s \in \mathcal{C}} \langle g^k, s \rangle$
8: **Update feasible solution**: $x^{k+1} \leftarrow x^k + \gamma_k(s^k - x^k)$.
9: **end for**
10: **return** x^{K+1}

Suppose we are at iteration k. Then as a prerequisite to solving the LMO (Linear Minimization Oracle) problem in Line 7 of Algorithm TUFW, we first need to compute an (accurate) estimate g^k of $\nabla F(x^k)$, which in turn means computing an (accurate) estimate g_i^k of $\nabla f_i(x^k)$ for $i \in [n]$. In an exact method we would simply compute $g_i^k \leftarrow \nabla f_i(x^k)$ and then compute $g^k \leftarrow \frac{1}{n}\sum_{i=1}^n g_i^k$, the cost of which grows at least as $O(np)$ operations. Here we instead consider, for each $i \in [n]$, computing the estimate g_i^k of $\nabla f_i(x^k)$ by using the second-order Taylor approximation of $\nabla f_i(x)$ around some point b_i, resulting in the estimate:

$$g_i^k \leftarrow \nabla f_i(b_i) + \nabla^2 f_i(b_i)(x^k - b_i) \quad \text{for } i \in [n] \ ,$$

which is Line 5 of TUFW. We refer to b_i as the "Taylor point" for the Taylor approximation of $\nabla f_i(\cdot)$, and $b_1, \ldots, b_n$ are the collection of Taylor points. The Taylor points for different observations might be different, namely $b_i \neq b_j$. Also, the Taylor point b_i for observation i will be *updated* from time to time over the course of the algorithm. In Algorithm 2.1 we initially set all Taylor points to be the initial point x^0, namely $b_i \leftarrow x^0$ for $i = 1, 2, \ldots, n$ in Line 2. Notice that there is an Taylor point b_i for each observation index $i \in [n]$.

In order for the Taylor points $b_1, \ldots, b_n$ to yield sufficiently accurate gradient estimates $g_1^k, \ldots, g_n^k$ of $\nabla g_1^k, \ldots, \nabla g_n^k$ at iteration k, we will need to update some/all of the Taylor points to the current feasible point iterate x^k for some subset $\mathcal{B}_k \subset [n]$ at iteration k of Algorithm TUFW. That is, we update $b_i \leftarrow x^k$ for $i \in \mathcal{B}_k$. This update is done in Line 4 of Algorithm TUFW.

The set $\mathcal{B}_k$ is the set of indices whose Taylor points are updated to x^k at iteration k, and $\mathcal{B}_k$ can be determined by any kind of **Rule**. As a trivial example, if one uses the

rule $\mathcal{B}_k = [n]$ for all k, then all Taylor points are updated to the current iterate x^k at each iteration k, in which case TUFW specializes to an instantiation of the traditional Frank-Wolfe method with exact gradient computation. As another example, one can use the rule that $\mathcal{B}_k$ is a randomly chosen subset of the indices of cardinality, say, 50, or $\lfloor n/10 \rfloor$, or perhaps is iteration dependent such as cardinality $\lfloor \ln(n)\sqrt{k} \rfloor$. As a third example, one can use the rule that $\mathcal{B}_k$ is chosen deterministically in a cyclic fashion with fixed batch-size B, so that $\mathcal{B}_1 = \{1, \ldots, B\}$, $\mathcal{B}_2 = \{B+1, \ldots, 2B\}$, etc., with modular arithmetic so that $\mathcal{B}_k \subset [n]$. Let $\beta_k := |\mathcal{B}_k|$ be the cardinality of $\mathcal{B}_k$. As we will see in succeeding sections, we can attain nearly-optimal complexity by developing batch-size rules for which β_k is a *decreasing* function of k, both in the convex and non-convex setting. This decreasing batch-size property is in stark contrast to other versions of Frank-Wolfe that use sampling of the observations to compute gradient estimates, wherein the batch-sizes grow with k, see [11, 29].

Going back to the description Algorithm 2.1, the Taylor-approximated (at Taylor point b_i) gradient of f_i is computed for each observation $i \in [n]$ in Line 5, and the estimate of the full gradient is computed in Line 6. Finally, the Linear Minimization Oracle (LMO) is called in Line 7 and the new iterate is computed in Line 8 using the step-size γ_k along the cord $[x^k, s^k]$.

Inspecting Algorithm 2.1, it appears that the computation of the gradient estimate g^k in Line 5 and Line 6 requires $O(np^2)$ flops. However, we now show how to do the computation of g^k in these steps in only $O((|\mathcal{B}_k|+1)p^2)$ flops by taking advantage of the "linear prediction" structure of the loss functions $f_i(x) := l_i(w_i^\top x)$ for each observation $i \in [n]$, and updating g^k based on the computed value of g^{k-1}.

PROPOSITION 2.1. *Let $\bar{\beta}_k := |\mathcal{B}_k|$. Then the updating of the second-order Taylor approximation model requires $O(\bar{\beta} p^2)$ flops and the computation of g^k can be done via a matrix-vector product, which takes at most $O(p^2)$ flops.*

Proof of Proposition 2.1. Consider iteration $k-1$ and let the Taylor points at this iteration be $b_1, \ldots, b_n$. Let us write out the computation of g^{k-1} as:

$$\begin{aligned} g^{k-1} &= \tfrac{1}{n}\sum_{i=1}^n l_i'(w_i^\top b_i) w_i + \tfrac{1}{n}\sum_{i=1}^n l_i''(w_i^\top b_i) w_i w_i^\top (x^{k-1} - b_i) \\ &= \tfrac{1}{n}\sum_{i=1}^n (v_i - t_i w_i^\top b_i) w_i + \left[\tfrac{1}{n}\sum_{i=1}^n t_i w_i w_i^\top\right] x^{k-1} = q_{k-1} + H_{k-1} x^{k-1} \ , \end{aligned}$$

where $v_i = l_i'(w_i^\top b_i)$, $t_i = l_i''(w_i^\top b_i)$ for $i \in [n]$, and $q_{k-1} = \frac{1}{n}\sum_{i=1}^n (v_i - t_i w_i^\top b_i) w_i$ and $H_{k-1} = \frac{1}{n}\sum_{i=1}^n t_i w_i w_i^\top$. We assume that we can store the values v_i, t_i for $i \in [n]$, as well as the vector q_{k-1} and Hessian matrix H_{k-1}.

At iteration k, let $\mathcal{B}_k$ be the set of observation indices i where we replace the Taylor point b_i with x^k. Using the overbar symbol "$\leftarrow$" to denote an update of a stored value, we have:

$$\begin{array}{llllll} \bar{b}_i = b_i & , & \bar{v}_i = v_i & , & \bar{t}_i = t_i & \text{for } i \notin \mathcal{B}_k \ , \quad \text{and} \\ \bar{b}_i = x^k & , & \bar{v}_i = l_i'(w_i^\top \bar{b}_i) & , & \bar{t}_i = l_i''(w_i^\top \bar{b}_i) & \text{for } i \in \mathcal{B}_k \ , \end{array}$$

and notice that these computations use $O(\bar{\beta}_k p)$ flops. We then compute g^k by first

updating q^{k-1} and H_{k-1} to q^k and H_k as follows:

$$\begin{aligned} q_k &= q_{k-1} + \tfrac{1}{n} \sum_{i \in \mathcal{B}_k} (\bar{v}_i - \bar{t}_i w_i^\top \bar{b}_i - v_i + t_i w_i^\top b_i) w_i \\ H_k &= H_{k-1} + \tfrac{1}{n} \sum_{i \in \mathcal{B}_k} (\bar{t}_i - t_i) w_i w_i^\top \\ g^k &= q_k + H_k x^k \ , \end{aligned} \tag{2.1}$$

where the first two steps of updating q_k and H_k in the Taylor approximation model require $O(\bar{\beta}_k p^2)$ flops and the last step of computing g^k needs only a matrix-vector product, which is at most $O(p^2)$ flops. After this, we replace the old values of b_i, v_i, and t_i with their new values $\bar{b}_i$, $\bar{v}_i$, and $\bar{t}_i$ for $i \in \mathcal{B}_k$, which requires $O(\bar{\beta}_k p)$ flops. ∎

Moreover, TUFW can be more computationally efficient by utilizing sparsity properties. Since the Frank-Wolfe method often promotes sparse structured solutions [13], the iterates x^k are typically sparse in certain settings. Furthermore for problems such as matrix completion [5, 25], the Hessian matrices are also highly sparse [25]. Therefore matrix-vector products and the updating of H_k can often be done with much less than $\mathcal{O}(p^2)$ flops by using sparse linear algebra. In addition, the batches $\mathcal{B}_k$ become empty with higher probability for large k, and in this case the q_k and H_k remain unchanged, and computing g^k can be simplified.

3. Convergence Guarantees for Convex Loss Functions. In this section we study convergence guarantees and overall complexity of Algorithm 2.1 for tackling problem (1.2) in the case when the scalar loss functions l_i are convex for all $i \in [n]$, which then implies that f_i and F are also convex functions. Later, in section 6, we will present numerical experiments where we compare different versions of TUFW applied to problems with convex loss functions.

Recall that in order to run Algorithm 2.1 we need to specify the step-sizes $\{\gamma_k\}_k$ and the **Rule** to construct the sets $\mathcal{B}_k$ of indices for Taylor-point updating in Line 4. Our first rule is designed for $\mathcal{B}_k$ to be comprised of $\beta_k := n/\sqrt{k}$ independently drawn samples from $[n]$ without replacement, for all $k \geq 1$. Since β_k is not in general an integer, then rather than ensuring $|\mathcal{B}_k| = \beta_k$, we will instead ensure that $\mathbb{E}|\mathcal{B}_k| = \beta_k$ by proceeding as follows.

DEFINITION 3.1. *Rule−SBD$\sqrt{k}$. Define $\beta_k := n/\sqrt{k}$ and $p_k := \beta_k - \lfloor \beta_k \rfloor$. Sample ξ_k from the Bernoulli distribution* $\mathrm{Ber}(p_k)$ *and then uniformly sample $\lfloor \beta_k \rfloor + \xi_k$ samples from $[n]$ without replacement. Then define $\mathcal{B}_k$ to be the set of these index samples.*

Note that by design of *Rule−SBD$\sqrt{k}$* it follows that $\mathbb{E}_{\xi_k}|\mathcal{B}_k| = \beta_k$. We call this rule "*Rule−SBD$\sqrt{k}$*" for "Stochastic Batch-size Decreasing at the rate $\sqrt{k}$," and we emphasize that the Taylor points are updated less often as k grows, which is in contrast to many other stochastic Frank-Wolfe methods that compute exact individual gradients more frequently as k grows, see [11, 29].

We first present a bound on the total number of flops used in the first k iterations of Algorithm 2.1 using *Rule−SBD$\sqrt{k}$*. We will presume that computing l_i' and l_i'' require $O(1)$ flops, and let fLMO denote the number of flops required by the LMO in Line 7 of Algorithm 2.1.

PROPOSITION 3.2. *Using Rule−SBD$\sqrt{k}$ and $k \geq 1$, the expected total number of flops used in the first k iterations of Algorithm 2.1 is $O\left(k \cdot (\mathrm{fLMO} + p^2) + \sqrt{k} \cdot np^2\right)$.*

Proof of Proposition 3.2. For the initial iteration of Algorithm TUFW the number of flops is $O(\text{fLMO} + np^2)$, and the expected total number of flops in the first k iterations is

$$O\left(k \cdot \text{fLMO} + np^2 + \sum_{i=1}^{k}(\beta_k + 1)p^2\right) = O\left(k \cdot \text{fLMO} + kp^2 + np^2 + \sum_{i=1}^{k}\frac{np^2}{\sqrt{i}}\right)$$
$$\leq O\left(k \cdot \left(\text{fLMO} + p^2\right) + np^2 + np^2 \cdot 2x^{\frac{1}{2}}\Big|_{x=1}^{k+1}\right) = O\left(k \cdot \left(\text{fLMO} + p^2\right) + np^2\sqrt{k}\right) ,$$

where the left-most term follows from Proposition 2.1 and the inequality uses a standard integral bound. ∎

Because the feasibility set $\mathcal{C}$ is bounded, the following measure of the range of the linear operator $W^\top$

$$D_q := \max_{x,y\in\mathcal{C}} \left\|W^\top(x-y)\right\|_q \tag{3.1}$$

is finite for all $q \in [1, \infty]$. We note that (3.1) is a common metric used in the literature of problem (1.2), see [18, 22].

The following theorem presents the computational guarantee for Algorithm 2.1 with *Rule−SBD*$\sqrt{k}$. The proof of this theorem, as well as the other ensuing results in this sectioin, are presented in subsection 3.1.

THEOREM 3.3. *Suppose that F is convex and Assumption 1.1 holds, and Algorithm 2.1 with Rule−SBD*$\sqrt{k}$ *is applied to the problem (1.2) with step-sizes defined by $\gamma_k := 2/(k+2)$ for all $k \geq 0$. Then for all $k \geq 1$ we have:*

$$\mathbb{E}[F(x^k) - F(x^\star)] \leq \frac{2LD_2^2 + 134\hat{L}D_1D_\infty^2}{n(k+1)} . \tag{3.2}$$

COROLLARY 3.4. *Under the hypotheses of Theorem 3.3, it holds that $\mathbb{E}[F(x^k) - F(x^\star)] \leq \varepsilon$ after at most*

$$\frac{2LD_2^2 + 134\hat{L}D_1D_\infty^2}{n\varepsilon}$$

iterations, and the total number of flops required is at most

$$O\left((\text{fLMO} + p^2)\left[\frac{LD_2^2 + \hat{L}D_1D_\infty^2}{n\varepsilon}\right] + np^2\left[\frac{\sqrt{LD_2^2 + \hat{L}D_1D_\infty^2}}{\sqrt{n}\sqrt{\varepsilon}}\right]\right) .$$

Let $\|\cdot\|$ be a given norm on R^p and let $\text{Diam}(\mathcal{C})$ denote the diameter of $\mathcal{C}$, namely $\text{Diam}(\mathcal{C}) := \max_{x,y\in\mathcal{C}} \|x-y\|$. Suppose that all feature vectors $w_1, w_2, \ldots$ lie in a bounded set $S \subset \{w \in \mathrm{R}^p : \|w\|_* \leq M\}$ regardless of the number of observations n, where M denotes the radius of the (dual norm) ball centered at the origin in $\mathbb{R}^p$ containing S. The following corollary follows from (3.2) using $D_2 \leq \sqrt{n}D_\infty \leq \sqrt{n}M \cdot \text{Diam}(\mathcal{C})$ and $D_1 \leq nD_\infty \leq nM \cdot \text{Diam}(\mathcal{C})$.

COROLLARY 3.5. *Under the boundedness of the feature vectors, the bound on the number of flops in Corollary 3.4 to obtain $\mathbb{E}[F(x^k) - F(x^\star)] \leq \varepsilon$ is*

$$O\left((\text{fLMO}+p^2)\cdot\frac{LM^2\cdot\text{Diam}(\mathcal{C})^2 + \hat{L}M^3\cdot\text{Diam}(\mathcal{C})^3}{\varepsilon} + np^2\cdot\frac{\sqrt{LM^2\cdot\text{Diam}(\mathcal{C})^2 + \hat{L}M^3\cdot\text{Diam}(\mathcal{C})^3}}{\sqrt{\varepsilon}}\right).$$

Whereas *Rule−SBD*$\sqrt{k}$ is stochastic, we now present the following deterministic rule which achieves nearly identical computational guarantees (to within a constant factor) but whose computational guarantees are deterministic.

DEFINITION 3.6. *Rule−DBD*$\sqrt{k}$. *For any* $k \geq 1$ *define:*

$$\mathcal{B}_k = \begin{cases} [n] & \text{if } \sqrt{k} \in \mathbb{N} \\ \emptyset & \text{if } \sqrt{k} \notin \mathbb{N} \end{cases}.$$

In *Rule−DBD*$\sqrt{k}$ we do not update any Taylor points unless $k = 1, 4, 9, 25, \ldots$, and for these values of k we update all n Taylor points. We call this approach "*Rule−DBD*$\sqrt{k}$" for "Deterministic Batch-frequency Decreasing at the rate $\sqrt{k}$," and we point out that for *Rule−DBD*$\sqrt{k}$ the Taylor points are updated less often as k grows (in a different way but with similar effect as in *Rule−SBD*$\sqrt{k}$).

Similar to the case of *Rule−SBD*$\sqrt{k}$, we have:

PROPOSITION 3.7. *Using Rule−DBD*$\sqrt{k}$ *and* $k \geq 1$, *the total number of flops used in the first* k *iterations of Algorithm 2.1 is* $O(k \cdot (\text{fLMO} + p^2) + \sqrt{k} \cdot np^2)$.

Proof of Proposition 3.7. The proof is almost identical to that of Proposition 3.2. For the initial iteration of Algorithm TUFW the number of flops is $O(\text{fLMO} + np^2)$, and the number of flops in the first k iterations is

$$O\left(k \cdot \text{fLMO} + np^2 + \textstyle\sum_{i=1}^{k}(\beta_k + 1)p^2\right) \quad \leq \quad O\left(k \cdot (\text{fLMO} + p^2) + np^2\sqrt{k}\right) \ ,$$

where the left-hand side follows from Proposition 2.1 and the inequality follows since $\sum_{i=1}^{k} \beta_k \leq n\sqrt{k}$. ∎

THEOREM 3.8. *Suppose that* F *is convex and Assumption 1.1 holds, and Algorithm 2.1 with Rule−DBD*$\sqrt{k}$ *is applied to the problem (1.2) with step-sizes defined by* $\gamma_k := 2/(k+2)$ *for all* $k \geq 0$. *Then for all* $k \geq 1$ *we have:*

$$F(x^k) - F(x^\star) \leq \frac{2LD_2^2 + 144\hat{L}D_3^3}{n(k+1)} \ . \tag{3.3}$$

COROLLARY 3.9. *Under the hypotheses of Theorem 3.8, it holds that* $F(x^k) - F(x^\star) \leq \varepsilon$ *after at most*

$$\frac{2LD_2^2 + 144\hat{L}D_3^3}{n\varepsilon}$$

iterations, and the total number of flops required is at most

$$O\left((\text{fLMO} + p^2)\left[\frac{LD_2^2 + \hat{L}D_3^3}{n\varepsilon}\right] + np^2\left[\frac{\sqrt{LD_2^2 + \hat{L}D_3^3}}{\sqrt{n}\sqrt{\varepsilon}}\right]\right) \ .$$

Remark 3.10. If we ignore the absolute constants then the bound in Theorem 3.8 is better that in Theorem 3.3 because $D_3^3 \leq D_1 D_\infty^2$.

Also similar to Corollary 3.5, we have the following corollary which shows that the joint dependence on n and ε is $O(n/\sqrt{\varepsilon})$:

COROLLARY 3.11. *Under the boundedness of the feature vectors, the bound on the number of flops in Corollary 3.9 to obtain* $F(x^k) - F(x^\star) \leq \varepsilon$ *is*

$$O\left((\text{fLMO}+p^2)\cdot\frac{LM^2 \cdot \text{Diam}(\mathcal{C})^2 + \hat{L}M^3 \cdot \text{Diam}(\mathcal{C})^3}{\varepsilon} + np^2 \cdot \frac{\sqrt{LM^2 \cdot \text{Diam}(\mathcal{C})^2 + \hat{L}M^3 \cdot \text{Diam}(\mathcal{C})^3}}{\sqrt{\varepsilon}}\right).$$

Table 1 shows a comparison of computational guarantees of the standard Frank-Wolfe method, the Constant batch-size Stochastic Frank-Wolfe (CSFW) method developed in [22], and the TUFW method with *Rule*$-SBD\sqrt{k}$ and *Rule*$-DBD\sqrt{k}$.

TABLE 1
Complexity bounds for different Frank-Wolfe methods to obtain an ε-optimal solution of ERM_ℓ with convex losses, under the boundedness assumption of the feature vectors. In the table $\varepsilon_0 := F(x^0) - F(x^\star)$, $c_1 := LM^2\mathrm{Diam}(\mathcal{C})^2$, and $c_2 := \hat{L}M^3\mathrm{Diam}(\mathcal{C})^3$.

Method	Optimality Metric	Overall Complexity
Rule$-SBD\sqrt{k}$ (Cor. 3.5)	$\mathbb{E}[F(x^k) - F(x^\star)] \leq \varepsilon$	$O\left((\text{fLMO} + p^2) \cdot \frac{c_1 + c_2}{\varepsilon} + np^2 \cdot \frac{\sqrt{c_1 + c_2}}{\sqrt{\varepsilon}}\right)$
Rule$-DBD\sqrt{k}$ (Cor. 3.11)	$F(x^k) - F(x^\star) \leq \varepsilon$	$O\left((\text{fLMO} + p^2) \cdot \frac{c_1 + c_2}{\varepsilon} + np^2 \cdot \frac{\sqrt{c_1 + c_2}}{\sqrt{\varepsilon}}\right)$
standard Frank-Wolfe ([13])	$F(x^k) - F(x^\star) \leq \varepsilon$	$O\left((\text{fLMO} + np) \cdot \frac{c_1}{\varepsilon}\right)$
CSFW ([22])	$\mathbb{E}[F(x^k) - F(x^\star)] \leq \varepsilon$	$O\left((n \cdot \text{fLMO} + np)\left[\frac{c_1}{\varepsilon} + \frac{\sqrt{\varepsilon_0}}{n\sqrt{\varepsilon}} + \frac{\sqrt{nc_1}}{\sqrt{\varepsilon}}\right]\right)$

Note in particular that with *Rule*$-SBD\sqrt{k}$ or *Rule*$-DBD\sqrt{k}$, the TUFW's joint dependence on n and ε is $O(n/\sqrt{\varepsilon})$, as compared to $O(n/\varepsilon)$ in [13, 22, 18], $O(1/\varepsilon^2)$ in [11, 29, 30, 9], or $O(1/\varepsilon^3)$ in [21]. Indeed, in the regime where ε is sufficiently small, the dominant term in the TUFW's complexity bound is the left term (which counts operations of the LMO), which is independent of n. In this regime the overall complexity is nearly optimal: the LMO complexity's dependence on ε is $O(1/\varepsilon)$, which is essentially the same as in the lower bound complexity result in [16]. The LMO complexity of TUFW differs from the lower bound only by the term involving $\hat{L}$, which is the Lipschitz constant of the second-order derivatives.

We end this subsection by considering the following rule which does no updates of the Taylor points beyond their initialization in Line 2 of TUFW.

DEFINITION 3.12. *Rule*$-\emptyset$. *Define $\mathcal{B}_0 = [n]$ and $\mathcal{B}_k := \emptyset$ for $k \geq 1$.*

Rule$-\emptyset$ is quite relevant in the special case when the loss functions l_i are all quadratic loss functions (as in linear regression, matrix completion, LASSO, etc.) in which case the Hessian $\nabla^2 F(x) = \nabla^2 F(x^0)$ for any x and so does not change. Therefore in this case $g_i^k = \nabla f_i(x^k)$ and $g^k = \nabla F(x^k)$ for all k. When using *Rule*$-\emptyset$, the computational cost of computing the initial individual gradients and the Hessian $\nabla^2 F(x^0)$ is only $O(np^2)$ flops, and the cost of calculating the (exact) gradient is the cost of a matrix-vector product, which is at most $O(p^2)$ flops per iteration.

3.1. Proofs of results in section 3. We will prove Theorem 3.3 as a consequence of the following sequences of lemmas, corollaries, and propositions.

LEMMA 3.13. *Under Assumption 1.1 for problem ERM_ℓ,*

$$F(x^{k+1}) \leq F(x^k) + \gamma_k \left\langle \nabla F(x^k), s^k - x^k \right\rangle + \gamma_k^2 L D_2^2 / 2n \tag{3.4}$$

holds for iteration k of Algorithm 2.1 for all $k \geq 0$.

Proof of Lemma 3.13. For iteration k it follows from (1.3) that

$$l_i(w_i^\top x^{k+1}) \leq l_i(w_i^\top x^k) + l_i'(w_i^\top x^k)(w_i^\top x^{k+1} - w_i^\top x^k) + L(w_i^\top x^{k+1} - w_i^\top x^k)^2/2 \,.$$

Then since $F(x^{k+1}) = \frac{1}{n}\sum_{i=1}^n l_i(w_i^\top x^{k+1})$, we have:

$$\begin{aligned} F\left(x^{k+1}\right) \leq\ & F(x^k) + \gamma_k \left\langle \nabla F(x^k), s^k - x^k \right\rangle + \gamma_k^2 L \left\| W^\top s^k - W^\top x^k \right\|_2^2 / 2n \\ \leq\ & F(x^k) + \gamma_k \left\langle \nabla F(x^k), s^k - x^k \right\rangle + \gamma_k^2 L D_2^2 / 2n \ , \end{aligned}$$

where the last inequality above uses (3.1). ∎

For any $k \geq 0$ and $i \in [n]$, we will be interested in the most recent iteration (up through k) at which Taylor point b_i was updated. This is defined as:

$$\tau_i^k := \max\{t : t \leq k \text{ and } i \in \mathcal{B}_t\} \ , \tag{3.5}$$

which means that in the first k iterations, τ_i^k is the latest iteration in which the Taylor point b_i was updated. In Algorithm 2.1, the relationship between τ_i^k and τ_i^{k-1} is then:

$$\tau_i^k = \begin{cases} k & \text{if } \ i \in \mathcal{B}_k \\ \tau_i^{k-1} & \text{if } \ i \notin \mathcal{B}_k \ . \end{cases} \tag{3.6}$$

Lemma 3.14. *Under Assumption 1.1 for the problem ERM$_\ell$, for any $u, v \in \mathcal{C}$,*

$$\left(\nabla F(x^k) - g^k\right)^\top (u - v) \leq \frac{\hat{L}}{2n} \sum_{i=1}^n \Big(\sum_{j=\tau_i^k}^{k-1} \gamma_j \left| w_i^\top (s^j - x^j) \right| \Big)^2 \cdot |w_i^\top (u - v)| \tag{3.7}$$

holds for iteration k of Algorithm 2.1 for all $k \geq 1$.

Proof of Lemma 3.14. It follows from the definition of τ_i^k in (3.5) that

$$\begin{aligned} & \left(\nabla F(x^k) - g^k\right)^\top (u - v) \\ &= \tfrac{1}{n} \sum_{i=1}^n \left(l_i'(w_i^\top x^k) - l_i'(w_i^\top x^{\tau_i^k}) - l_i''(w_i^\top x^{\tau_i^k})(w_i^\top x^k - w_i^\top x^{\tau_i^k}) \right) \cdot w_i^\top (u - v) \\ &\leq \tfrac{\hat{L}}{2n} \sum_{i=1}^n |w_i^\top x^k - w_i^\top x^{\tau_i^k}|^2 \cdot |w_i^\top (u - v)| = \frac{\hat{L}}{2n} \sum_{i=1}^n \Big(\sum_{j=\tau_i^k}^{k-1} w_i^\top (x^{j+1} - x^j) \Big)^2 \cdot |w_i^\top (u - v)| \\ &\leq \tfrac{\hat{L}}{2n} \sum_{i=1}^n \Big(\sum_{j=\tau_i^k}^{k-1} \left| w_i^\top (x^{j+1} - x^j) \right| \Big)^2 \cdot |w_i^\top (u - v)| \\ &= \tfrac{\hat{L}}{2n} \sum_{i=1}^n \Big(\sum_{j=\tau_i^k}^{k-1} \left| \gamma_j w_i^\top (s^j - x^j) \right| \Big)^2 \cdot |w_i^\top (u - v)| \end{aligned} \tag{3.8}$$

where the first inequality uses (1.4). ∎

Because $|w_i^\top (u - v)| \leq D_\infty$, Lemma 3.14 implies the following corollary.

Corollary 3.15.

$$\left(\nabla F(x^k) - g^k\right)^\top (u - v) \leq \frac{\hat{L} D_\infty}{2n} \sum_{i=1}^n \Big(\sum_{j=\tau_i^k}^{k-1} \gamma_j \left| w_i^\top (s^j - x^j) \right| \Big)^2 \ . \tag{3.9}$$

Lemma 3.16. *For any $k \in \mathbb{N}$ the following inequality holds:*

$$\sqrt{\lfloor k/2 \rfloor + 1} \exp(2\sqrt{\lfloor k/2 \rfloor + 1} - 2\sqrt{k+1}) \leq \min\{3(k+2)^{-\frac{1}{2}}, 10(k+2)^{-1}\}. \tag{3.10}$$

Proof. Since $h(x) := \sqrt{x}$ is a concave function on $\mathbb{R}_+$, we have $h(k+2) \leq h(k+1) + h'(k+1)$, which yields $\sqrt{k+1} \geq \sqrt{k+2} - \frac{1}{2\sqrt{k+1}} \geq \sqrt{k+2} - \sqrt{2}/4$ when $k \geq 1$. Furthermore $\sqrt{\lfloor k/2 \rfloor + 1} \leq \frac{\sqrt{2}}{2}\sqrt{k+2}$, and therefore

$$\sqrt{\lfloor k/2 \rfloor + 1}\exp(2\sqrt{\lfloor k/2 \rfloor + 1} - 2\sqrt{k+1}) \leq \frac{\sqrt{2}e^{\frac{\sqrt{2}}{2}}}{2}(k+2)^{\frac{1}{2}}\exp(-(2-\sqrt{2})(k+2)^{\frac{1}{2}}) \ .$$

Next note that the function $h(x) := e^{-x}x^t$ on the domain $x \in (0, \infty)$ for $t > 0$ attains its maximal value at $x = t$. In particular for $t = 2$ and $t = 3$, respectively, we conclude that $e^{-x} \leq 4e^{-2}x^{-2}$ and $e^{-x} \leq 27e^{-3}x^{-3}$ (respectively), and thus $\exp(-(2-\sqrt{2})(k+2)^{\frac{1}{2}}) \leq 4e^{-2}(2-\sqrt{2})^{-2}(k+2)^{-1}$ and $\exp(-(2-\sqrt{2})(k+2)^{\frac{1}{2}}) \leq 27e^{-3}(2-\sqrt{2})^{-3}(k+2)^{-3/2}$ (respectively). Therefore, the left-hand side of (3.10) is bounded by $2\sqrt{2}e^{\frac{\sqrt{2}}{2}-2}(2-\sqrt{2})^{-2}(k+2)^{-\frac{1}{2}}$ and $\frac{27}{2}\sqrt{2}e^{\frac{\sqrt{2}}{2}-3}(2-\sqrt{2})^{-3}(k+2)^{-1}$, respectively, which is less than $3(k+2)^{-\frac{1}{2}}$ and $10(k+2)^{-1}$, respectively. ∎

Let F^* denote the optimal objective function value of ERM_ℓ and let $\varepsilon_k := F(x^k) - F^*$ denote the optimality gap at iteration k.

LEMMA 3.17. *Under Assumption 1.1 for problem ERM_ℓ, suppose that F is convex, and let the step-sizes be given by $\gamma_k = 2/(k+2)$ for all $k \geq 0$. Then for all $k \geq 1$ we have*

$$\varepsilon_k \leq \frac{2LD_2^2}{n(k+1)} + \frac{2}{(k)(k+1)}\sum_{t=1}^{k} t\left(\nabla F(x^{t-1}) - g^{t-1}\right)^\top (s^{t-1} - x^\star) \ . \tag{3.11}$$

Proof of Lemma 3.17. Utilizing Lemma 3.13 we have

$$\begin{aligned} F(x^{k+1}) \leq\ & F(x^k) + \gamma_k\langle \nabla F(x^k), s^k - x^k\rangle + \gamma_k^2 LD_2^2/2n \\ =\ & F(x^k) + \gamma_k\langle \nabla F(x^k) - g^k, s^k - x^k\rangle + \gamma_k\langle g^k, s^k - x^k\rangle + \gamma_k^2 LD_2^2/2n \\ \leq\ & F(x^k) + \gamma_k\langle \nabla F(x^k) - g^k, s^k - x^k\rangle + \gamma_k\langle g^k, x^\star - x^k\rangle + \gamma_k^2 LD_2^2/2n \\ =\ & F(x^k) + \gamma_k\langle \nabla F(x^k) - g^k, s^k - x^\star\rangle + \gamma_k\langle \nabla F(x^k), x^\star - x^k\rangle + \gamma_k^2 LD_2^2/2n \\ \leq\ & F(x^k) + \gamma_k\langle \nabla F(x^k) - g^k, s^k - x^\star\rangle + \gamma_k(F(x^\star) - F(x^k)) + \gamma_k^2 LD_2^2/2n \ , \end{aligned} \tag{3.12}$$

where the second inequality uses $s^k \in \arg\min_{s\in\mathcal{C}}(g^k)^\top s$ from Line 7 of the algorithm, and the third inequality uses the gradient inequality applied to F. Subtracting $F^\star$ from both sides of the above inequality chain, we arrive at:

$$\varepsilon_{k+1} \leq (1-\gamma_k)\varepsilon_k + \gamma_k\left(\nabla F(x^k) - g^k\right)^\top (s^k - x^\star) + \gamma_k^2 LD_2^2/2n \ . \tag{3.13}$$

Multiplying both side by $(k+1)(k+2)$ and telescoping the inequalities yields:

$$(k+1)(k+2)\varepsilon_{k+1} \leq \frac{2(k+1)LD_2^2}{n} + \sum_{t=0}^{k} 2(t+1)(\nabla F(x^t) - g^t)^\top (s^t - x^\star) \ , \tag{3.14}$$

which yields (3.11) after rearranging and re-indexing the counter t. ∎

LEMMA 3.18. *Let the series $\{d_{ij}\}_{i\in[n], j\in\mathbb{N}}$ be given, and suppose that there exists M_1 and M_∞ for which $\sup_j \sum_{i=1}^n d_{ij} \leq M_1$ and $\sup_{i,j} d_{ij} \leq M_\infty$. Then for Algorithm 2.1 using Rule−SBD$\sqrt{k}$ and step-sizes $\gamma_k := 2/(k+2)$ for all $k \geq 0$, it holds that:*

$$\mathbb{E}\Big[\sum_{i=1}^{n}\Big(\sum_{j=\tau_i^k}^{k-1}\gamma_j d_{ij}\Big)\Big] \leq \frac{6M_1}{\sqrt{k+2}} \quad \text{and} \quad \mathbb{E}\Big[\sum_{i=1}^{n}\Big(\sum_{j=\tau_i^k}^{k-1}\gamma_j d_{ij}\Big)^2\Big] \leq \frac{134M_1M_\infty}{k+2} \ . \tag{3.15}$$

Before proving the above lemma, we first introduce the following proposition.

PROPOSITION 3.19. *For integers v and k satisfying $1 \le v \le k$ it holds that:*

$$\sum_{i=1}^{v}\prod_{j=i}^{k}\left(1-\tfrac{1}{\sqrt{j}}\right) < (\sqrt{v+1})\exp(2\sqrt{v+1}-2\sqrt{k+1}) \ . \tag{3.16}$$

Proof of Proposition 3.19. For convenience let us define $\rho_j := 1-\frac{1}{\sqrt{j}}$ for $j \ge 1$, and define the function $h(x) := \exp(2\sqrt{x}-2\sqrt{k+1})$ for $x \ge 0$. We will first prove the following inequality:

$$\prod_{j=i}^{k}\left(1-\tfrac{1}{\sqrt{j}}\right) \le h(i) \quad \text{for all } i \in [k] \ . \tag{3.17}$$

For $i=1$ the left side above is 0 (because $\rho_1 = 0$ and hence $\prod_{j=1}^{k}\rho_j = 0$), and the right side is $h(1) > 0$, so (3.17) is true for $i=1$. For $i \ge 2$ we have:

$$\begin{aligned}\prod_{j=i}^{k}\rho_j &= \exp\left(\sum_{j=i}^{k}\ln(1-\tfrac{1}{\sqrt{j}})\right) \le \exp\left(-\sum_{j=i}^{k}\tfrac{1}{\sqrt{j}}\right)\\ &\le \exp\Big(-\int_{i}^{k+1} x^{-\frac{1}{2}}dx\Big) = \exp(2\sqrt{i}-2\sqrt{k+1}) = h(i) \ ,\end{aligned}$$

where the first inequality follows from the fact that $\log(1-x) \le -x$ for any $0 \le x < 1$, and the second inequality is an integral bound. This establishes (3.17), whereby we have

$$\begin{aligned}\sum_{i=1}^{v}\prod_{j=i}^{k}\rho_j \le \sum_{i=1}^{v}h(i) &\le \int_{1}^{v+1} h(x)\ dx = (\sqrt{x}-\tfrac{1}{2})\exp(2\sqrt{x}-2\sqrt{k+1})\Big|_{1}^{v+1}\\ &= (\sqrt{v+1}-\tfrac{1}{2})\exp(2\sqrt{v+1}-2\sqrt{k+1}) - (\tfrac{1}{2})\exp(2-2\sqrt{k+1})\\ &< (\sqrt{v+1}-\tfrac{1}{2})\exp(2\sqrt{v+1}-2\sqrt{k+1}) \ ,\end{aligned}$$

where the second inequality is an integral bound using the fact that $h(x)$ is an increasing function. ∎

Proof of Lemma 3.18. First of all, let us define $G_k^i := \left(\sum_{j=\tau_i^k}^{k-1}\gamma_j d_{ij}\right)$ and $Z_k^i := \left(\sum_{j=\tau_i^k}^{k-1}\gamma_j d_{ij}\right)^2$ for $i \in [n]$, and also $G_k := \sum_{i=1}^{n}G_k^i$ and $Z_k := \sum_{i=1}^{n}Z_k^i$. Notice in G_k^i and Z_k^i that the randomness lies only in τ_i^k, which in turn depends on the update sets $\mathcal{B}_0, \mathcal{B}_1, \ldots, \mathcal{B}_k$.

We first analyze G_k^i and G_k. We note that $G_k^i = 0$ when $i \in \mathcal{B}_k$, which happens with probability β_k/n. Because of the relationship between $\{\tau_i^k\}$ and $\{\tau_i^{k-1}\}$ in (3.6), we further note that with probability $1-\beta_k/n$ that $i \notin \mathcal{B}_k$ in which case it follows that $G_k^i = G_{k-1}^i + \gamma_{k-1}d_{i,k-1}$. Defining $\rho_k := 1-\frac{1}{\sqrt{k}} = 1-\beta_k/n$, we have

$$\mathbb{E}[G_k] = \sum_{i=1}^{n}\mathbb{E}[G_k^i] = \rho_k\sum_{i=1}^{n}\mathbb{E}[G_{k-1}^i+\gamma_{k-1}d_{i,k-1}] \le \rho_k\left(\mathbb{E}[G_{k-1}]+\gamma_{k-1}M_1\right) \ , \tag{3.18}$$

where the inequality uses $\sup_j\sum_{i=1}^{n}d_{ij} \le M_1$. Recurrently applying (3.18) yields:

$$\mathbb{E}[G_k] \le \rho_k(\mathbb{E}[G_{k-1}]+\gamma_{k-1}M_1) \le \cdots \le \sum_{i=1}^{k}\Big(\prod_{j=i}^{k}\rho_j\Big)\gamma_{i-1}M_1 + \Big(\prod_{j=1}^{k}\rho_j\Big)\mathbb{E}[G_0] \ . \tag{3.19}$$

Noting that $G_0 = 0$ (since all Taylor points are set to x^0 in Line 2 of Algorithm 2.1) and using $\gamma_{i-1} = \frac{2}{i+1}$ it follows that:

$$\begin{aligned}\mathbb{E}[G_k] &\leq \sum_{i=1}^{\lfloor k/2 \rfloor} \Big(\prod_{j=i}^{k} \rho_j\Big) \frac{2M_1}{i+1} + \sum_{i=\lfloor k/2 \rfloor+1}^{k} \Big(\prod_{j=i}^{k} \rho_j\Big) \frac{2M_1}{i+1} \\ &\leq \sum_{i=2}^{\lfloor k/2 \rfloor} \Big(\prod_{j=i}^{k} \rho_j\Big) \frac{2M_1}{3} + \sum_{i=\lfloor k/2 \rfloor+1}^{k} \Big(\prod_{j=i}^{k} \rho_j\Big) \frac{2M_1}{\lfloor k/2 \rfloor + 2} \,,\end{aligned} \tag{3.20}$$

where the second inequality uses the fact that $\rho_1 = 0$ and hence $\prod_{j=1}^{k} \rho_j = 0$. By Proposition 3.19 we have:

$$\sum_{i=1}^{\lfloor k/2 \rfloor} \Big(\prod_{j=i}^{k} \rho_j\Big) \frac{2M_1}{3} \leq \sqrt{\lfloor k/2 \rfloor + 1} \exp\Big(2\sqrt{\lfloor k/2 \rfloor + 1} - 2\sqrt{k+1}\Big) \frac{2M_1}{3} \leq \frac{2M_1}{\sqrt{k+2}} \,, \tag{3.21}$$

where the second inequality above uses Lemma 3.16. Similarly for the second term in the rightmost side of (3.20) we have:

$$\sum_{i=\lfloor k/2 \rfloor+1}^{k} \Big(\prod_{j=i}^{k} \rho_j\Big) \frac{2M_1}{\lfloor k/2 \rfloor + 2} \leq \sum_{i=1}^{k} \Big(\prod_{j=i}^{k} \rho_j\Big) \frac{2M_1}{\lfloor k/2 \rfloor + 2} \leq \frac{\sqrt{k+1} \cdot 2M_1}{\lfloor k/2 \rfloor + 2} \leq \frac{4M_1}{\sqrt{k+2}} \,, \tag{3.22}$$

where the second inequality uses Proposition 3.19. Substituting (3.21) and (3.22) back into (3.20) yields the first part of (3.15), namely:

$$\mathbb{E}[G_k] \leq 6M_1/\sqrt{k+2} \,. \tag{3.23}$$

We employ a similar proof approach to establish the second part of (3.15). We note that $Z_k^i = 0$ when $i \in \mathcal{B}_k$, which happens with probability β_k/n. Because of the relationship between $\{\tau_i^k\}$ and $\{\tau_i^{k-1}\}$ in (3.6), we further note that with probability $1 - \beta_k/n$ that $i \notin \mathcal{B}_k$ in which case we have

$$\begin{aligned}Z_k^i &= (G_{k-1}^i + \gamma_{k-1} d_{i,k-1})^2 = Z_{k-1}^i + 2G_{k-1}^i \gamma_{k-1} d_{i,k-1} + (\gamma_{k-1} d_{i,k-1})^2 \\ &\leq Z_{k-1}^i + 2M_\infty \gamma_{k-1} G_{k-1}^i + \gamma_{k-1}^2 M_\infty d_{i,k-1} \,.\end{aligned}$$

Therefore

$$\begin{aligned}\mathbb{E}[Z_k] &= \sum_{i=1}^{n} \mathbb{E}[Z_k^i] \leq \rho_k \sum_{i=1}^{n} \mathbb{E}[Z_{k-1}^i + 2M_\infty \gamma_{k-1} G_{k-1}^i + \gamma_{k-1}^2 M_\infty d_{i,k-1}] \\ &\leq \rho_k \left(\mathbb{E}[Z_{k-1}] + 2M_\infty \gamma_{k-1} \mathbb{E}[G_{k-1}] + \gamma_{k-1}^2 M_\infty M_1\right) \\ &\leq \rho_k \left(\mathbb{E}[Z_{k-1}] + \frac{24 M_\infty M_1}{(k+1)\sqrt{k+1}} + \frac{4M_\infty M_1}{(k+1)^2}\right) \leq \rho_k \left(\mathbb{E}[Z_{k-1}] + \frac{28 M_\infty M_1}{(k+1)\sqrt{k+1}}\right) \,,\end{aligned} \tag{3.24}$$

where the second inequality uses $\sup_j \sum_{i=1}^{n} d_{ij} \leq M_1$, and the third inequality uses (3.23). Recurrently applying (3.24) yields:

$$\mathbb{E}[Z_k] \leq \rho_k \left(\mathbb{E}[Z_{k-1}] + \frac{28 M_1 M_\infty}{(k+1)^{3/2}}\right) \leq \cdots \leq \sum_{i=1}^{k} \Big(\prod_{j=i}^{k} \rho_j\Big) \frac{28 M_1 M_\infty}{(i+1)^{3/2}} \,. \tag{3.25}$$

Therefore

$$\begin{aligned}\mathbb{E}[Z_k] &\leq \sum_{i=1}^{\lfloor k/2 \rfloor} \Big(\prod_{j=i}^{k} \rho_j\Big) \frac{28 M_1 M_\infty}{(i+1)^{3/2}} + \sum_{i=\lfloor k/2 \rfloor+1}^{k} \Big(\prod_{j=i}^{k} \rho_j\Big) \frac{28 M_1 M_\infty}{(i+1)^{3/2}} \\ &\leq \sum_{i=1}^{\lfloor k/2 \rfloor} \Big(\prod_{j=i}^{k} \rho_j\Big) \frac{28 M_1 M_\infty}{3\sqrt{3}} + \sum_{i=\lfloor k/2 \rfloor+1}^{k} \Big(\prod_{j=i}^{k} \rho_j\Big) \frac{28 M_1 M_\infty}{(\lfloor k/2 \rfloor + 2)^{3/2}} \,.\end{aligned} \tag{3.26}$$

By Proposition 3.19 we have

$$\sum_{i=1}^{\lfloor k/2 \rfloor} \Big(\prod_{j=i}^{k} \rho_j \Big) \frac{28 M_1 M_\infty}{3\sqrt{3}} \leq \sqrt{\lfloor k/2 \rfloor + 1} \exp\big(2\sqrt{\lfloor k/2 \rfloor + 1} - 2\sqrt{k+1}\big) \cdot \frac{28 M_1 M_\infty}{(3\sqrt{3})} \leq \frac{54 M_1 M_\infty}{k+2} \,, \tag{3.27}$$

where the second inequality above uses Lemma 3.16. Similarly for the second term in the rightmost side of (3.26) we have:

$$\sum_{i=\lfloor k/2 \rfloor + 1}^{k} \Big(\prod_{j=i}^{k} \rho_j \Big) \frac{28 M_1 M_\infty}{(\lfloor k/2 \rfloor + 2)^{3/2}} \leq \sum_{i=1}^{k} \Big(\prod_{j=i}^{k} \rho_j \Big) \frac{28 M_1 M_\infty}{(\lfloor k/2 \rfloor + 2)^{3/2}} \leq \sqrt{k+1} \cdot \frac{28 M_1 M_\infty}{(\lfloor k/2 \rfloor + 2)^{3/2}} \leq \frac{80 M_1 M_\infty}{k+2} \,, \tag{3.28}$$

where the second inequality uses Proposition 3.19. Substituting (3.27) and (3.28) back into (3.26) yields the second part of (3.15), namely:

$$\mathbb{E}[Z_k] \leq \frac{134 M_1 M_\infty}{k+2} \,. \tag{3.29}$$

∎

Proof of Theorem 3.3. From Lemma 3.17 it holds that:

$$\varepsilon_k \leq \frac{2LD_2^2}{n(k+1)} + \frac{2}{k(k+1)} \sum_{t=1}^{k} t \left(\nabla F(x^{t-1}) - g^{t-1} \right)^\top \left(s^{t-1} - x^\star \right) , \tag{3.30}$$

and from Corollary 3.15 it follows that:

$$\varepsilon_k \leq \frac{2LD_2^2}{n(k+1)} + \frac{\hat{L} D_\infty}{nk(k+1)} \sum_{t=1}^{k} t \sum_{i=1}^{n} \Big(\sum_{j=\tau_i^{t-1}}^{t-2} \gamma_j \left| w_i^\top (s^j - x^j) \right| \Big)^2 .$$

Taking expectations on both sides then yields

$$\mathbb{E}[\varepsilon_k] \leq \frac{2LD_2^2}{n(k+1)} + \frac{\hat{L} D_\infty}{nk(k+1)} \sum_{t=1}^{k} t \, \mathbb{E}\Big[\sum_{i=1}^{n} \Big(\sum_{j=\tau_i^{t-1}}^{t-2} \gamma_j \left| w_i^\top (s^j - x^j) \right| \Big)^2 \Big] . \tag{3.31}$$

Now define $d_{ij} := \left| w_i^\top (s^j - x^j) \right|$ and $M_1 := D_1$ and $M_\infty := D_\infty$. Then since $\sup_j \sum_{i=1}^n d_{ij} = \sup_j \sum_{i=1}^n \left| w_i^\top (s^j - x^j) \right| \leq D_1 = M_1$ and $\sup_{i,j} d_{ij}$ is equal to $\sup_{i,j} \left| w_i^\top (s^j - x^j) \right|$ which is bounded by $D_\infty = M_\infty$, we can apply Lemma 3.18 to (3.31) and obtain

$$\mathbb{E}[\varepsilon_k] \leq \frac{2LD_2^2}{n(k+1)} + \frac{\hat{L} D_\infty}{nk(k+1)} \sum_{t=1}^{k} t \, \frac{134 D_1 D_\infty}{t+1} \leq \frac{2LD_2^2}{n(k+1)} + \frac{134 \hat{L} D_1 D_\infty^2}{n(k+1)} , \tag{3.32}$$

which demonstrates (3.2) . ∎

We next proceed towards the proof of Theorem 3.8. We first prove the following lemma.

LEMMA 3.20. *In any iteration k of TUFW outlined in Algorithm 2.1 with $\gamma_k := 2/(k+2)$ and Rule−DBD$\sqrt{k}$, for problem setup (1.2), it holds that*

$$\sum_{t=1}^{k} t \sum_{i=1}^{n} \Big(\sum_{j=\tau_i^{t-1}}^{t-2} \gamma_j \left| w_i^\top (z^j - y^j) \right| \Big)^2 \left| w_i^\top (z^{t-1} - x^\star) \right| \leq 144 k D_3^3 \tag{3.33}$$

for any $y^j, z^j \in \mathcal{C}$, $j = 1, \ldots, k$.

Proof of Lemma 3.20. Notice that in *Rule−DBD*$\sqrt{k}$, $\tau_i^k = (\lfloor\sqrt{k}\rfloor)^2$, so $k - \tau_i^k = k - (\lfloor\sqrt{k}\rfloor)^2 \le \lfloor 2\sqrt{k}\rfloor$, where the latter follows from squaring the inequality $\lfloor\sqrt{k}\rfloor \ge \sqrt{k} - 1$ and rearranging. Therefore,

$$\begin{aligned}
&\sum_{t=1}^{k} t \sum_{i=1}^{n} \Big(\sum_{j=\tau_i^{t-1}}^{t-2} \gamma_j \left| w_i^\top (z^j - y^j) \right| \Big)^2 |w_i^\top (z^{t-1} - x^\star)| \\
\le &\sum_{t=1}^{k} t \sum_{i=1}^{n} \Bigg(\sum_{j=t-1-\lfloor 2\sqrt{t-1}\rfloor}^{t-2} \frac{2\left| w_i^\top (z^j - y^j) \right|}{t+1-\lfloor 2\sqrt{t-1}\rfloor} \Bigg)^2 |w_i^\top (z^{t-1} - x^\star)| \qquad (3.34) \\
\le &\sum_{t=1}^{k} \frac{4t\lfloor 2\sqrt{t-1}\rfloor}{(t+1-\lfloor 2\sqrt{t-1}\rfloor)^2} \sum_{j=t-1-\lfloor 2\sqrt{t-1}\rfloor}^{t-2} \sum_{i=1}^{n} \left| w_i^\top (z^j - y^j) \right|^2 |w_i^\top (z^{t-1} - x^\star)|
\end{aligned}$$

where the first inequality is due to $\tau_i^{t-1} \ge t - 1 - \lfloor 2\sqrt{t-1}\rfloor$ and the second inequality is due to the fact that for any vector $a \in \mathbb{R}^d$ we have $\|a\|_1^2 \le d\|a\|_2^2$. In the right-hand side of (3.34), we have

$$\frac{4t\lfloor 2\sqrt{t-1}\rfloor}{(t+1-\lfloor 2\sqrt{t-1}\rfloor)^2} \le \frac{8t^{3/2}}{t^2/9} = 72t^{-1/2} \ , \qquad (3.35)$$

where the inequality involving the denominators above follows from verifying $t + 1 - \lfloor 2\sqrt{t-1}\rfloor \ge t/3$ via the quadratic formula. Additionally notice that for any vectors $a, b \in \mathbb{R}^d$ and letting A denote the diagonal matrix whose diagonal elements are $a_1, a_2, \ldots, a_d$, we have $\sum_{i=1}^d a_i^2 b_i = e^\top A^2 b \le \|A^2 e\|_u \|b\|_v = \|a\|_{2u}^2 \|b\|_v$ for any $u,\ v$ such that $u,\ v \ge 1$ and $\frac{1}{u} + \frac{1}{v} = 1$. Therefore $\sum_{i=1}^n \left| w_i^\top (z^j - y^j) \right|^2 |w_i^\top (z^{t-1} - x^\star)| \le D_{2u}^2 D_v$ and it then follows that

$$\sum_{j=t-1-\lfloor 2\sqrt{t-1}\rfloor}^{t-2} \sum_{i=1}^{n} \left| w_i^\top (z^j - y^j) \right|^2 |w_i^\top (z^{t-1} - x^\star)| \le \lfloor 2\sqrt{t-1}\rfloor D_{2u}^2 D_v \le 2\sqrt{t} D_{2u}^2 D_v \ . \qquad (3.36)$$

The inequality (3.33) then follows by setting $u = 3/2$, $v = 3$, and substituting (3.35) and (3.36) into (3.34). ∎

Proof of Theorem 3.8. Similar to the proof of Theorem 3.3, according to Lemmas 3.14 and 3.17,

$$\varepsilon_k \le \frac{2LD_2^2}{n(k+1)} + \frac{\hat{L}}{nk(k+1)} \sum_{t=1}^{k} t \sum_{i=1}^{n} \Big(\sum_{j=\tau_i^{t-1}}^{t-2} \gamma_j \left| w_i^\top (s^j - x^j) \right| \Big)^2 |w_i^\top (s^{t-1} - x^\star)| \ .$$

Next apply Lemma 3.20 to the above inequality, which yields (3.3). ∎

4. Convergence Guarantees for Non-convex Loss Functions. In this section we study convergence rates and overall complexity of Algorithm 2.1 for tackling problem (1.2) in the case that the scalar loss functions l_i are not necessarily convex. Later, in section 6, we will present experimental results comparing different versions of TUFW applied to problems with non-convex loss functions.

Recall that the Frank-Wolfe gap $\mathcal{G}(x)$ at x was defined in (1.7), and when F is convex $\mathcal{G}(x^k)$ is an upper bound on the optimality gap at x^k, see [13] for example. The Frank-Wolfe gap does not necessarily bound the optimality gap in the non-convex setting; nevertheless $\mathcal{G}(x)$ is always nonnegative and $\mathcal{G}(x) = 0$ if and only if x is a stationary point of ERM_ℓ, and for this reason $\mathcal{G}(x)$ is often used as a measure of

non-stationarity at x, see [15, 23, 29, 22]. Hence for ERM_ℓ with non-convex objective function F, we say that a point $x \in \mathcal{C}$ is an ε-stationarity point of ERM_ℓ if $\mathcal{G}(x) \leq \varepsilon$.

Recall also that in order to run Algorithm 2.1, we need to specify the step-sizes $\{\gamma_k\}_k$ and the **Rule** for constructing the sets $\mathcal{B}_k$ of indices for updating the Taylor-points in Line 4 of Algorithm 2.1. Let K denote the number of iterations that we intend to run. In a somewhat similar spirit as the rule *Rule−SBD*$\sqrt{k}$ for convex losses, we present the following stochastic updating rule that is designed for $\mathcal{B}_k$ to be comprised of $\beta_k := n/\sqrt[4]{K}$ independently drawn samples from $[n]$ without replacement, for all $k \geq 1$. If β_k is not integral, we will instead ensure $\mathbb{E}|\mathcal{B}_k| = \beta_k$ by using a similar approach to *Rule−SBD*$\sqrt{k}$ which uses a Bernoulli random variable. With this in mind, we formally define the following **Rule**:

Definition 4.1. *Rule−SBD*$\sqrt[4]{K}$. *For a fixed value of $K \geq 1$, and for all $k \geq 1$ define $\beta_k := \beta := n/\sqrt[4]{K}$ and $p_k := p := \beta_k - \lfloor \beta_k \rfloor$. Sample ξ_k from the Bernoulli distribution Ber(p_k) and then uniformly sample $\lfloor \beta_k \rfloor + \xi_k$ samples from $[n]$ without replacement. Then define $\mathcal{B}_k$ to be the set of these index samples.*

Note that by design of *Rule−SBD*$\sqrt[4]{K}$ it also follows that $\mathbb{E}_{\xi_k}|\mathcal{B}_k| = \beta_k$. We call this rule "*Rule−SBD*$\sqrt[4]{K}$" for "Stochastic Batch-size Decreasing at the rate $\sqrt[4]{K}$," because the Taylor points are updated less often as the number of overall iterations grow. Next we present a bound on the total number of flops used in K iterations of Algorithm 2.1 using *Rule−SBD*$\sqrt[4]{K}$.

Proposition 4.2. *Using Rule−SBD*$\sqrt[4]{K}$ *and $K \geq 1$ iterations, the expected total number of flops used in Algorithm 2.1 is $O(K \cdot (\text{fLMO} + p^2) + K^{3/4} \cdot np^2)$.*

Proof of Proposition 4.2. For the initial iteration of Algorithm 2.1 the number of flops is $O(\text{fLMO} + np^2)$, and the expected number of flops in each iteration thereafter is $O(\text{fLMO} + p^2 + np^2/\sqrt[4]{K})$. Summing over the K iterations yields the result. ∎

The following theorem presents a computational guarantee for computing an ε-stationarity point of ERM_ℓ using Algorithm 2.1 with *Rule−SBD*$\sqrt[4]{K}$. The proof of this theorem and other ensuing results in the section are presented in subsection 4.1.

Theorem 4.3. *Suppose that Assumption 1.1 holds and F is not necessarily convex, and Algorithm 2.1 with Rule−SBD*$\sqrt[4]{K}$ *is applied to the problem (1.2) with step-sizes defined by $\gamma_k := \gamma := 1/\sqrt{K+1}$ for all $k \geq 0$, where $K \geq 1$ is given. Then:*

$$\mathbb{E}[\min_{k \in \{0,\ldots,K\}} \mathcal{G}(x^k)] \leq \sum_{k=0}^{K} \frac{\mathbb{E}[\mathcal{G}(x^k)]}{K+1} \leq \frac{F(x^0) - F(x^*)}{\sqrt{K+1}} + \frac{3\hat{L}D_1D_\infty^2 + LD_2^2}{2n\sqrt{K+1}} \,. \tag{4.1}$$

The following corollary presents an alternative version of Theorem 4.3 by randomly choosing an iteration index $\hat{k} \in [K]$, and shows that the joint dependence on n and ε is $O(n/\varepsilon^{3/2})$ under the hypothesis that all feature vectors $w_1, w_2, \ldots$ lie in a bounded set $S \subset \{w \in \mathrm{R}^p : \|w\|_* \leq M\}$.

Corollary 4.4. *Under the boundedness of the feature vectors, the bound on the number of flops required to obtain $\mathbb{E}_{\hat{k} \sim \mathcal{U}([K])}\mathbb{E}[\mathcal{G}(x^{\hat{k}})] \leq \varepsilon$ is*

$$O\left((\text{fLMO} + p^2) \left[\frac{\left((F(x^0) - F(x^*)) + LM^2 \cdot \text{Diam}(\mathcal{C})^2 + \hat{L}M^3 \cdot \text{Diam}(\mathcal{C})^3 \right)^2}{\varepsilon^2} \right] \right.$$
$$\left. + np^2 \left[\frac{\left((F(x^0) - F(x^*)) + LM^2 \cdot \text{Diam}(\mathcal{C})^2 + \hat{L}M^3 \cdot \text{Diam}(\mathcal{C})^3 \right)^{3/2}}{\varepsilon^{3/2}} \right] \right) \,.$$

In the expectation $\mathbb{E}_{\hat{k}\sim\mathcal{U}([K])}\mathbb{E}[\mathcal{G}(x^{\hat{k}})]$ in Corollary 4.4, the randomness comes from both the random sampling at each iteration k to construct the update batches $\mathcal{B}_k$ in *Rule*$-SBD\sqrt[4]{K}$ as well as the random sampling of the iteration index $\hat{k}$ from the uniform distribution on $[K]$.

In a similar spirit as *Rule*$-DBD\sqrt{k}$, we also have the following deterministic rule.

DEFINITION 4.5. *Rule*$-DBD\sqrt[4]{K}$. *For a fixed $K \geq 1$, and for any $k \geq 1$, define:*

$$\mathcal{B}_k = \begin{cases} [n] & \text{if } k/\lfloor\sqrt[4]{K}\rfloor \in \mathbb{N} \\ \emptyset & \text{if } k/\lfloor\sqrt[4]{K}\rfloor \notin \mathbb{N} \end{cases}.$$

In *Rule*$-DBD\sqrt[4]{K}$ we do not update any Taylor points unless k is an integer times $\lfloor\sqrt[4]{K}\rfloor$, and for these values of k we update all n Taylor points. We refer the interested reader to the technical report [28] for computational guarantees for *Rule*$-DBD\sqrt[4]{K}$.

Table 2 shows a comparison of the computational guarantees of the standard Frank-Wolfe method and TUFW with *Rule*$-SBD\sqrt[4]{K}$.

TABLE 2

Complexity bounds for different Frank-Wolfe methods to obtain an ε-stationary solution of ERM$_\ell$ with non-convex losses, under the boundedness assumption of the feature vectors. In the table $\varepsilon_0 := F(x^0) - F(x^\star)$, $c_1 := LM^2\text{Diam}(\mathcal{C})^2$, and $c_2 := \hat{L}M^3\text{Diam}(\mathcal{C})^3$.

Method	Optimality Metric	Overall Complexity
Rule$-SBD\sqrt[4]{K}$ (Corollary 4.4)	$\mathbb{E}_{\hat{k}\sim\mathcal{U}([K])}\mathbb{E}[\mathcal{G}(x^{\hat{k}})] \leq \varepsilon$	$O\left((\text{fLMO} + p^2)\cdot\frac{(\varepsilon_0+c_1+c_2)^2}{\varepsilon^2} + np^2\cdot\frac{(\varepsilon_0+c_1+c_2)^{3/2}}{\varepsilon^{3/2}}\right)$
standard Frank-Wolfe ([15])	$\mathbb{E}_{\hat{k}\sim\mathcal{U}([K])}[\mathcal{G}(x^{\hat{k}})] \leq \varepsilon$	$O\left((\text{fLMO} + np)\cdot\frac{(\varepsilon_0+c_1)^2}{\varepsilon^2}\right)$

Note in particular that with *Rule*$-SBD\sqrt[4]{K}$, the joint dependence on n and ε is $O(n/\varepsilon^{3/2})$, as compared to $O(n^{2/3}/\varepsilon^2)$ in [23], $O(n^{1/2}/\varepsilon^2)$ in [29], and $O(1/\varepsilon^3)$ in [30, 9]. In the regime where ε is sufficiently small, the dominant term in the complexity bound is the left-most term which counts the operations of the LMO, which is $O(1/\varepsilon^2)$ (independent of n), and is essentially the same as the lower bound complexity in [16]. Because of this, we say that TUFW's overall complexity is nearly optimal. We also mention that the joint dependence on n and ε for CASPIDERRG in [25] is $O(n^{3/4}/\varepsilon^{3/2})$; however CASPIDERG requires access to the exact full gradients periodically, and in our computational experience we found that CASPIDERG was not computationally viable relative to other stochastic Frank-Wolfe methods.

Finally, we point out that when the loss functions l_i are all quadratic loss functions, the Hessian $\nabla^2 F(x)$ is constant for for all x, and we can use *Rule*$-\emptyset$ which is the rule that does not update any of the Taylor points.

4.1. Proofs of results in section 4. Our proofs will use Lemmas 3.13 and 3.14 since they do not assume convexity of F. In addition our proofs will rely on the following sequences of lemmas. We will use ε_0 to denote $F(x^0) - F(x^\star)$ in the rest of this section.

LEMMA 4.6. *Under Assumption 1.1 for problem (1.2), if we use the step-size $\gamma_k := \gamma := 1/\sqrt{K+1}$ in Algorithm 2.1, then*

$$\sum_{k=0}^{K}\frac{\mathcal{G}(x^k)}{K+1} \leq \frac{\varepsilon_0}{\sqrt{K+1}} + \frac{1}{K+1}\sum_{k=0}^{K}(\nabla F(x^k) - g^k)^\top(s^k - \bar{s}^k) + \frac{LD_2^2}{2n\sqrt{K+1}}. \tag{4.2}$$

Proof of Lemma 4.6. For any $k \geq 0$ we have:

$$\begin{aligned} F(x^{k+1}) \leq\ & F(x^k) + \gamma_k(\nabla F(x^k))^\top(s^k - x^k) + \gamma_k^2 L D_2^2/2n \\ =\ & F(x^k) + \gamma_k(g^k)^\top(s^k - x^k) + \gamma_k(\nabla F(x^k) - g^k)^\top(s^k - x^k) + \gamma_k^2 L D_2^2/2n \\ \leq\ & F(x^k) + \gamma_k(g^k)^\top(\bar{s}^k - x^k) + \gamma_k(\nabla F(x^k) - g^k)^\top(s^k - x^k) + \gamma_k^2 L D_2^2/2n \\ =\ & F(x^k) - \gamma_k \mathcal{G}(x^k) + \gamma_k(g^k - \nabla F(x^k))^\top(\bar{s}^k - s^k) + \gamma_k^2 L D_2^2/2n \ , \end{aligned} \tag{4.3}$$

where $\bar{s}^k \in \arg\min_{s\in\mathcal{C}}(\nabla F(x^k)^\top s$, and the first inequality uses Lemma 3.13, and the second inequality uses the fact that $s^k \in \arg\min_{s\in\mathcal{C}}(g^k)^\top s$. Substituting the value of γ_k in the statement of the lemma yields after rearranging terms:

$$\frac{\mathcal{G}(x^k)}{\sqrt{K+1}} \leq F(x^k) - F(x^{k+1}) + \frac{1}{\sqrt{K+1}}(\nabla F(x^k) - g^k)^\top(s^k - \bar{s}^k) + \frac{LD_2^2}{2n(K+1)} \ . \tag{4.4}$$

The inequality (4.2) then follows by first summing the inequalities (4.4) for $k \in \{0,\ldots,K\}$ and dividing by $\sqrt{K+1}$, and noting that $F(x^0) - F(x^{K+1}) \leq \varepsilon_0$. ∎

LEMMA 4.7. *Let the series $\{d_{ij}\}_{i\in[n],j\in\mathbb{N}}$ be given, and suppose that there exists M_1 and M_∞ for which $\sup_j \sum_{i=1}^n d_{ij} \leq M_1$ and $\sup_{i,j} d_{ij} \leq M_\infty$. Then for Algorithm 2.1 using Rule−SBD $\sqrt[4]{K}$ with step-sizes $\gamma_k \equiv \gamma := 1/\sqrt{K+1}$, it holds that:*

$$\mathbb{E}\Big[\sum_{i=1}^n \sum_{j=\tau_i^k}^{k-1} \gamma_j d_{ij}\Big] \leq \frac{M_1}{(K+1)^{1/4}} \quad \textit{and} \quad \mathbb{E}\Big[\sum_{i=1}^n \Big(\sum_{j=\tau_i^k}^{k-1} \gamma_j d_{ij}\Big)^2\Big] \leq \frac{3M_1M_\infty}{\sqrt{K+1}} \ . \tag{4.5}$$

Proof of Lemma 4.7. Similar to the proof of Lemma 3.18, let us first define $G_k^i := \Big(\sum_{j=\tau_i^k}^{k-1} \gamma_j d_{ij}\Big)$ and $Z_k^i := \Big(\sum_{j=\tau_i^k}^{k-1} \gamma_j d_{ij}\Big)^2$ for $i \in [n]$, and also $G_k := \sum_{i=1}^n G_k^i$ and $Z_k := \sum_{i=1}^n Z_k^i$. First notice in G_k^i and Z_k^i that the randomness lies only in τ_i^k, which in turn depends on the update sets $\mathcal{B}_0, \mathcal{B}_1, \ldots, \mathcal{B}_k$. We note that $G_k^i = 0$ when $i \in \mathcal{B}_k$, which happens with probability $\beta/n = 1/\sqrt[4]{K}$. Because of the relationship between $\{\tau_i^k\}$ and $\{\tau_i^{k-1}\}$ in (3.6), we further note that with probability $1 - \beta/n$ that $i \notin \mathcal{B}_k$ in which case it follows that $G_k^i = G_{k-1}^i + \gamma_{k-1} d_{i,k-1} = G_{k-1}^i + \gamma d_{i,k-1}$. Defining $\rho := 1 - \beta/n = 1 - 1/\sqrt[4]{K}$, we have

$$\mathbb{E}[G_k] = \sum_{i=1}^n \mathbb{E}[G_k^i] = \rho \sum_{i=1}^n \mathbb{E}[G_{k-1}^i + \gamma d_{i,k-1}] \leq \rho\,(\mathbb{E}[G_{k-1}] + \gamma M_1) \ , \tag{4.6}$$

where the inequality uses $\sup_j \sum_{i=1}^n d_{ij} \leq M_1$. Now (4.6) can be rearranged to yield:

$$\mathbb{E}[G_k] - \frac{\rho\gamma M_1}{1-\rho} \leq \rho\Big(\mathbb{E}[G_{k-1}] - \frac{\rho\gamma M_1}{1-\rho}\Big) \leq \cdots \leq \rho^k\Big(\mathbb{E}[G_0] - \frac{\rho\gamma M_1}{1-\rho}\Big) \leq 0 \ , \tag{4.7}$$

where the last inequality uses $G_0 = 0$ (since all Taylor points are set to x^0 in Line 2 of Algorithm 2.1). We therefore have:

$$\mathbb{E}[G_k] \leq \frac{\rho\gamma M_1}{1-\rho} = \frac{(1 - K^{-1/4})(K+1)^{-1/2} M_1}{K^{-1/4}} \leq \frac{M_1}{(K+1)^{1/4}} \ , \tag{4.8}$$

which proves the first inequality of (4.5).

We proceed similarly to establish the second inequality of (4.5). We note that $Z_k^i = 0$ when $i \in \mathcal{B}_k$, which happens with probability $\beta/n = 1/\sqrt[4]{K}$. Because of the relationship between $\{\tau_i^k\}$ and $\{\tau_i^{k-1}\}$ in (3.6), we further note that with probability

$\rho = 1 - \beta/n$ that $i \notin \mathcal{B}_k$ in which case it follows that $Z_k^i = (G_{k-1}^i + \gamma d_{i,k-1})^2$. We thus have

(4.9)
$$\begin{aligned}
&\mathbb{E}[Z_k] = \sum_{i=1}^{n} \mathbb{E}[Z_k^i] \\
&= \rho \sum_{i=1}^{n} \mathbb{E}[(G_{k-1}^i + \gamma d_{i,k-1})^2] = \rho \sum_{i=1}^{n} \Big(\mathbb{E}[Z_{k-1}^i] + 2\gamma \mathbb{E}[G_{k-1}^i] d_{i,k-1} + \gamma^2 (d_{i,k-1})^2] \Big) \\
&\leq \rho \left(\mathbb{E}[Z_{k-1}] + 2\gamma \mathbb{E}[G_{k-1}] M_\infty + \gamma^2 (M_\infty M_1) \right) \leq \rho \left(\mathbb{E}[Z_{k-1}] + 2\gamma \frac{M_1 M_\infty}{(K+1)^{1/4}} + \gamma^2 M_\infty M_1 \right) \\
&= \rho \left(\mathbb{E}[Z_{k-1}] + 2 \frac{M_1 M_\infty}{(K+1)^{3/4}} + \frac{M_\infty M_1}{K+1} \right) \leq \rho \left(\mathbb{E}[Z_{k-1}] + \frac{3 M_1 M_\infty}{(K+1)^{3/4}} \right) .
\end{aligned}$$

Now (4.9) can be rearranged to yield:

$$\begin{aligned}
\mathbb{E}[Z_k] - \frac{3 M_1 M_\infty}{(K+1)^{3/4}} \cdot \frac{\rho}{1-\rho} &\leq \rho \Big(\mathbb{E}[Z_{k-1}] - \frac{3 M_1 M_\infty}{(K+1)^{3/4}} \cdot \frac{\rho}{1-\rho} \Big) \\
&\leq \cdots \leq \rho^k \Big(\mathbb{E}[Z_0] - \frac{3 M_1 M_\infty}{(K+1)^{3/4}} \cdot \frac{\rho}{1-\rho} \Big) \leq 0 ,
\end{aligned} \tag{4.10}$$

where the last inequality uses $Z_0 = 0$ (since all Taylor points are set to x^0 in Line 2 of Algorithm 2.1), from which it follows that

$$\mathbb{E}[Z_k] \leq \frac{3 M_1 M_\infty}{(K+1)^{3/4}} \cdot \frac{\rho}{1-\rho} \leq \frac{3 M_1 M_\infty}{\sqrt{K+1}} , \tag{4.11}$$

which proves the second inequality of (4.5). ∎

Proof of Theorem 4.3. Notice that the first inequality of (4.1) follows from the chain

$$\mathbb{E}[\min_{k \in \{0,\ldots,K\}} \mathcal{G}(x^k)] \leq \min_{k \in \{0,\ldots,K\}} \mathbb{E}[\mathcal{G}(x^k)] \leq \sum_{k=0}^{K} \frac{\mathbb{E}[\mathcal{G}(x^k)]}{K+1} .$$

From Lemma 4.6 we have:

$$\sum_{k=0}^{K} \frac{\mathcal{G}(x^k)}{K+1} \leq \frac{2n\varepsilon_0 + L D_2^2}{2n\sqrt{K+1}} + \frac{1}{K+1} \sum_{k=0}^{K} (\nabla F(x^k) - g^k)^\top (s^k - \bar{s}^k) , \tag{4.12}$$

and applying Corollary 3.15 we obtain:

$$\sum_{k=0}^{K} \frac{\mathcal{G}(x^k)}{K+1} \leq \frac{2n\varepsilon_0 + L D_2^2}{2n\sqrt{K+1}} + \frac{\hat{L} D_\infty}{2n(K+1)} \sum_{k=0}^{K} \sum_{i=1}^{n} \Big(\sum_{j=\tau_i^k}^{k-1} \gamma_j \left| w_i^\top (s^j - x^j) \right| \Big)^2 . \tag{4.13}$$

Taking expectations on both sides, it follows that:

$$\sum_{k=0}^{K} \frac{\mathbb{E}[\mathcal{G}(x^k)]}{K+1} \leq \frac{2n\varepsilon_0 + L D_2^2}{2n\sqrt{K+1}} + \frac{\hat{L} D_\infty}{2n(K+1)} \sum_{k=0}^{K} \mathbb{E}\Big[\sum_{i=1}^{n} \Big(\sum_{j=\tau_i^k}^{k-1} \gamma_j \left| w_i^\top (s^j - x^j) \right| \Big)^2 \Big]. \tag{4.14}$$

Now define $d_{ij} := \left| w_i^\top (s^j - x^j) \right|$ and $M_1 := D_1$ and $M_\infty := D_\infty$. Then since $\sup_j \sum_{i=1}^n d_{ij}$ is equal to $\sup_j \sum_{i=1}^n \left| w_i^\top (s^j - x^j) \right|$ which is at most $D_1 = M_1$, and $\sup_{i,j} d_{ij}$ is equal to $\sup_{i,j} \left| w_i^\top (s^j - x^j) \right|$ which is at most $D_\infty = M_\infty$, we can apply the second inequality of Lemma 4.7 to (4.14) and we obtain the second inequality of (4.1). ∎

5. Extensions. We briefly describe two extensions of our results, namely (i) adaptive step-sizes, and (ii) amending TUFW for the general ERM problem (1.1). Regarding adaptive step-sizes, note that the results in the last two sections are based on non-adaptive step-size rules, in part because the methods only work with approximate gradients and so objective function improvement is more difficult to prove. However, because of the extra smoothness of the second derivatives of the loss functions l_i, it turns out that one can build an accurate-enough local quadratic model of each l_i which will make an adaptive step-size work well. Consider the following adaptive step-size for TUFW:

$$\tilde{\gamma}_k := \begin{cases} \min\left\{\gamma_k, \frac{(g^k)^\top(x^k-s^k)}{(s^k-x^k)^\top H_k(s^k-x^k)}\right\} & \text{if } (s^k-x^k)^\top H_k(s^k-x^k) > 0\ , \\ \gamma_k & \text{if } (s^k-x^k)^\top H_k(s^k-x^k) \le 0\ , \end{cases} \tag{5.1}$$

where H_k is defined in (2.1) and γ_k is the standard step-size. Our computational results in section 6 show the huge advantage of (5.1) over the comparable non-adaptive step-size, and computational guarantees for (5.1) are presented in the technical report [28].

TUFW can also be amended to tackle the general ERM problem (1.1) with amended assumptions that the gradient $\nabla f_i(\cdot)$ is L-Lipschitz continuous on $\mathcal{C}$ and the Hessian $\nabla^2 f_i(x)$ is $\hat{L}$-Lipschitz continuous on $\mathcal{C}$. Computational guarantees for TUFW for solving (1.1) are presented in [28].

6. Computational Experiments. In this section we present the results of computational experiments where we compare Algorithm TUFW (with the four different batch construction Rules developed earlier) and five other methods, namely the standard Frank-Wolfe method [6, 13] plus related methods in the recent literature, as follows:

- (FW) – the standard Frank-Wolfe method, see [6, 13]
- (FW-ada) – the standard Frank-Wolfe method with adaptive step-size [3], which is defined by $\gamma_k := \min\{1, \mathcal{G}(x^k)/(L\|x^k - s^k\|^2)\}$, where L is the Lipschitz constant
- (SPIDER-FW) – the Frank-Wolfe method with stochastic path-integrated differential estimator technique developed in [29]
- (CSFW) – the constant batch-size stochastic Frank-Wolfe method developed in [22], and
- (CASPIDERG) – curvature-aided stochastic path-integrated differential estimator gradient developed in [25].

We tested these methods on both convex and non-convex instances of ERM$_\ell$. We first conducted computational tests on problems with convex losses, using instances of the ℓ_1-constrained logistic regression problem (LR):

$$\text{LR:} \quad \min_x \ F(x) := \tfrac{1}{n}\sum_{i=1}^n \ln(1+\exp(-y_i w_i^\top x)) \quad \text{s.t. } \|x\|_1 \le \lambda\ , \tag{6.1}$$

where the ℓ_1-ball constraint is designed to induce sparsity and/or regularization. We chose instances of LR from LIBSVM [2] for which the number of observations $n \gg 0$ but not so large as to render the problem intractable. This yielded 12 instances from LIBSVM, namely `a1a`, `a2a`, `a8a`, `a9a`, `w1a`, `w2a`, `w7a`, `w8a`, `svmguide3`, `phishing`, `ijcnn1`, and `covtype`, whose dimensions n and p are displayed in the third and forth columns of Table 3. The value of λ was specified by 5-fold cross-validation of each instance. However, we used the same cross-validated λ for `a1a`, `a2a`, `a8a`, and `a9a`,

and similarly for `w1a`, `w2a`, `w7a`, for `w8a`. (Separately, we also tested the methods on larger feasible regions by enlarging each instance's feasbile region using $\lambda' := 100\lambda$.) All methods were implemented and run in Python on the MIT Engaging Cluster. To ensure fairness, our implementations did not use sparse linear algebra for any methods, nor did we implement the techniques discussed in the last paragraph of section 2.

We measured the performance of methods on problems with convex losses using the Frank-Wolfe gap $\mathcal{G}(x^k)$ defined in (1.7), since $\mathcal{G}(x^k)$ is a (conservative) upper bound on the optimality gap, namely $F(x^k) - F^* \leq \mathcal{G}(x^k)$ (see [13]), and the optimal value F^* is not known. We computed $\mathcal{G}(x^k)$ separately and offline after-the-fact.

Note that *Rule*$-SBD\sqrt{k}$ and *Rule*$-SBD\sqrt[4]{K}$ for TUFW require uniform sampling of $\bar{\beta}_k := |\mathcal{B}_k|$ indices from $[n]$ without replacement. These indices are then used to perform matrix and vector operations on the corresponding indexed columns of W. Because matrix and vector operations involving random indices are inherently inefficient in practice, we instead implemented *Rule*$-SBD\sqrt{k}$ and *Rule*$-SBD\sqrt[4]{K}$ by uniformly sampling a starting index $\hat{i} \sim \mathcal{U}([n])$ and then assigning the other $|\bar{\beta}_k| - 1$ indices successively to construct $\mathcal{B}_k := \{\hat{i}, \hat{i}+1, \ldots, \hat{i}+\bar{\beta}_k - 1\}$ (modulo n). This modification does not affect any of the theoretical convergence guarantees because (3.18) and (3.24) still hold and thus Lemma 3.18 is still true, and similarly (4.6) and (4.9) still hold and thus Lemma 4.7 is still true.

Figure 1 shows the performance of TUFW with *Rule*$-SBD\sqrt{k}$ and *Rule*$-DBD\sqrt{k}$, and the standard Frank-Wolfe method, on the dataset `a9a` (which we found to be representative of typical performance). The methods were run both with and without adaptive step-sizes. Notice that the TUFW methods significantly outperform the Frank-Wolfe method. *Rule*$-DBD\sqrt{k}$ using adaptive step-sizes significantly outperforms the other methods in terms of overall runtime and has similar LMO dependence as *Rule*$-SBD\sqrt{k}$. The second-best performance is from TUFW with *Rule*$-SBD\sqrt{k}$ with adaptive step-sizes. As these results are typical, we will focus on *Rule*$-SBD\sqrt{k}$ and *Rule*$-DBD\sqrt{k}$ with adaptive step-sizes in further comparisons with other Frank-Wolfe methods.

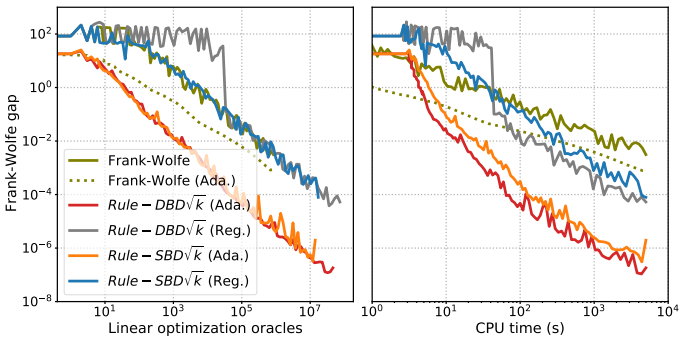

FIG. 1. *Performance of TUFW with Rule*$-SBD\sqrt{k}$ *and Rule*$-DBD\sqrt{k}$ *and the standard Frank-Wolfe method, with and without adaptive step-sizes, on the logistic regression problem (6.1), on the dataset* `a9a`.

In Table 3 we compare the CPU runtimes (in seconds) of the TUFW methods using *Rule*$-SBD\sqrt{k}$ and *Rule*$-DBD\sqrt{k}$, with FW, FW-ada, SPIDER-FW, and CSFW. (We did not include CASPIDERG in these tests as it is not designed for convex objective functions.) The batch-size of CSFW was set to $n/100$, which is exactly as in the implementation in [22]. For each method and each dataset we ran 10 trials, and

we report the average of these trials. We only report average runtimes that did not exceed 5000 seconds, otherwise the entry is blank. In each row of the table the bold number highlights the best runtime among the six methods. The last column of the table reports the speed-up of TUFW (the best of *Rule−SBD*$\sqrt{k}$ and *Rule−DBD*$\sqrt{k}$) over the best of the other four methods. Speed-up values larger than 1 are shown in bold. We observe from Table 3 that the TUFW method – using either *Rule−SBD*$\sqrt{k}$ or *Rule−DBD*$\sqrt{k}$, significantly outperforms the other methods on almost all problem instances and almost all target optimality tolerances. Furthermore, as the optimality tolerance ε decreases and/or as the number of observations n increase, the advantage of TUFW methods over the other methods becomes even more evident. Additional computational experiments with TUFW are presented in [28].

TABLE 3

Comparison of average CPU runtimes (in seconds) required to achieve $\mathcal{G}(x^k) \leq \varepsilon$ for methods on the logistic regression problem (6.1) with λ chosen by cross-validation. (A blank indicates the method used more than 5000 seconds.)

ε	dataset	n	p	*Rule−SBD*$\sqrt{k}$	*Rule−DBD*$\sqrt{k}$	FW	FW-ada	SPIDER-FW	CSFW	Speed-up
1e-1	a1a	1605	123	0.73	**0.37**	2.02	1.18	348.58	5.03	**3.20**
1e-3	a1a	1605	123	14.40	**5.14**	138.87	380.10		160.02	**27.02**
1e-5	a1a	1605	123	3029.61	**1034.66**	2956.50				**2.86**
1e-1	a2a	2265	123	1.02	**0.52**	3.03	1.55	358.72	5.47	**2.97**
1e-3	a2a	2265	123	21.24	**8.16**	197.80	472.72		167.66	**20.54**
1e-5	a2a	2265	123	2039.75	**631.23**					
1e-1	a8a	22696	123	7.75	**4.66**	78.18	12.83	443.85	15.22	**2.75**
1e-3	a8a	22696	123	51.29	**23.71**		3178.02		1250.12	**52.72**
1e-5	a8a	22696	123	420.64	**174.87**					
1e-1	a9a	32561	123	11.26	**6.80**	94.50	21.00	440.73	19.82	**2.91**
1e-3	a9a	32561	123	65.25	**31.22**		3844.13		1660.13	**53.17**
1e-5	a9a	32561	123	490.36	**197.15**					
1e-1	w1a	2477	300	10.45	**5.71**	17.51	331.22	3566.83	44.39	**3.07**
1e-3	w1a	2477	300	51.51	**21.93**	327.38			1219.03	**14.93**
1e-5	w1a	2477	300	1743.08	**541.71**					
1e-1	w2a	3470	300	13.11	**6.80**	37.34	574.42		54.06	**5.50**
1e-3	w2a	3470	300	128.14	**50.77**	369.82			1207.01	**7.28**
1e-5	w2a	3470	300		**1422.93**					
1e-1	w7a	24692	300	88.86	**55.43**	541.10			129.19	**2.33**
1e-3	w7a	24692	300	320.17	**178.87**				4413.77	**24.68**
1e-5	w7a	24692	300	3106.98	**1516.83**					
1e-1	w8a	49749	300	174.89	**114.66**	1198.35			212.80	**1.86**
1e-3	w8a	49749	300	553.94	**355.23**					
1e-5	w8a	49749	300		**2707.92**					
1e-1	svmguide3	1243	22	0.09	**0.02**	1.35	0.31	21.46	2.06	**13.43**
1e-3	svmguide3	1243	22	8.01	**2.64**	53.64	175.49		48.85	**18.53**
1e-5	svmguide3	1243	22	1132.50	**410.63**	629.39			1484.23	**1.53**
1e-7	svmguide3	1243	22							
1e-1	phishing	11055	68	0.47	0.38	0.01	0.71	0.01	**0.01**	0.02
1e-3	phishing	11055	68	2.81	1.24	0.67	201.65	**0.32**	0.53	0.26
1e-5	phishing	11055	68	45.36	**16.10**	47.01		1391.81	43.70	**2.71**
1e-7	phishing	11055	68	2478.92	**812.38**	3370.12				**4.15**
1e-1	ijcnn1	49990	22	0.85	0.24	1.21	9.02	**0.22**	0.92	0.91
1e-3	ijcnn1	49990	22	4.15	**0.86**	54.72	565.14	193.34	71.65	**63.40**
1e-5	ijcnn1	49990	22	7.67	**1.66**		2322.76			**1402.31**
1e-7	ijcnn1	49990	22	10.20	**2.32**		3910.71			**1688.65**
1e-1	covtype	581012	54	54.62	29.76	143.55	123.79	**5.47**	18.91	0.18
1e-3	covtype	581012	54	552.47	**231.70**	2964.49		843.60	690.78	**2.98**
1e-5	covtype	581012	54		**3777.45**					

We next tested the methods on non-convex instances of ERM$_\ell$, using instances of the ℓ_1-constrained binary classification problem with least-squares thresholding using the sigmoid function, namely:

$$\text{BCP:} \quad \min_x \ \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \frac{1}{1+\exp(-w_i^\top x)}\right)^2 \quad \text{s.t. } \|x\|_1 \leq \lambda\ , \tag{6.2}$$

using the same 12 datasets from LIBSVM as described earlier. Recall that the Frank-

Wolfe gap $\mathcal{G}(x)$ is a measure of non-stationarity, see [15, 23, 29, 22]. In the non-convex setting the design of most Frank-Wolfe methods involves setting the number of iterations K in advance and using a fixed step-size (usually $1/\sqrt{K+1}$), and the convergence rate for the Frank-Wolfe gap is typically measured using the expectation of $\mathcal{G}(x^{\hat{k}})$ where $\hat{k}$ is uniformly sampled over $\{0,\dots,K\}$. In our experiments we therefore compute the average Frank-Wolfe gap by computing $\sum_{i=0}^{K}\mathcal{G}(x^i)/(K+1)$ and then averaging this value over 5 independent trials. In order to save on computation when $K \geq 100n$, we only compute and record the Frank-Wolfe gaps every 200 iterations. Just as in the convex case, we computed the Frank-Wolfe gaps separately and offline after-the-fact.

For each $K \in \times\{10, 20, 10\times 2^2, \dots, 10\times 2^{20}\}$, we ran 5 independent trials of the methods on the 12 datasets. Figure 2 shows the performance of TUFW with *Rule−SBD* $\sqrt[4]{K}$ and *Rule−DBD* $\sqrt[4]{K}$, and the standard Frank-Wolfe method, on the dataset `a9a`, both with and without adaptive step-sizes. Each dot in the right-most subfigure corresponds to the average Frank-Wolfe gap, and the average runtime (over 5 trials) using a particular value of K as described above. Similar to the convex case, the TUFW methods significantly outperform the Frank-Wolfe method. *Rule−DBD* $\sqrt[4]{K}$ using adaptive step-sizes exhibits the best performance over the other methods in terms of overall runtime and has nearly identical LMO dependence as *Rule−SBD* $\sqrt[4]{K}$. The second-best performance is from TUFW with *Rule−SBD* $\sqrt[4]{K}$ with adaptive step-sizes. We focus on *Rule−SBD* $\sqrt[4]{K}$ and *Rule−DBD* $\sqrt[4]{K}$ with adaptive step-sizes in further comparisons with other Frank-Wolfe methods.

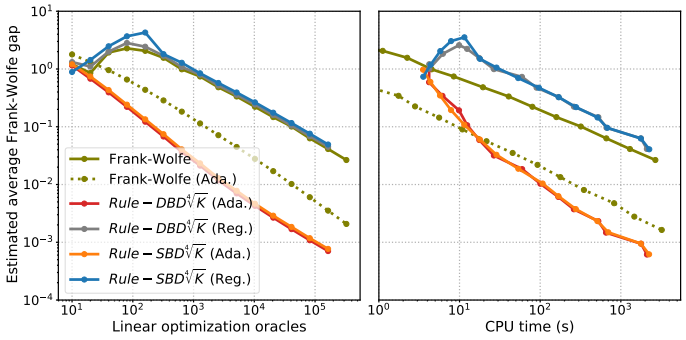

FIG. 2. *Performance of TUFW with Rule−SBD* $\sqrt[4]{K}$ *and Rule−DBD* $\sqrt[4]{K}$ *and the standard Frank-Wolfe method, with and without adaptive step-sizes, on the binary classification problem* (6.2)*, on the dataset* `a9a`.

In Table 4 we compare the CPU runtimes (in seconds) of the TUFW methods using *Rule−SBD* $\sqrt[4]{K}$ and *Rule−DBD* $\sqrt[4]{K}$, with FW, FW-ada, SPIDER-FW, and CASPIDERG. CASPIDERG calculates full gradients every $n^{1/4}K^{1/4}$ iterations and its batch-size is set to $\max\{n^{3/4}K^{-1/4}, 1\}$. (We did not include CSFW in these tests as it is not designed for non-convex objective functions.) All methods were run for each of the 21 different values of K and were terminated if they exceeded 5000 seconds. Further details on these experiments are presented in [28]. In each row of the table the bold number highlights the best runtime among the six methods. The last column of the table reports the speed-up of TUFW (the best of *Rule−SBD* $\sqrt[4]{K}$ and *Rule−DBD* $\sqrt[4]{K}$) over the best of the other four methods. Speed-up values larger than 1 are shown in bold. Notice from Table 4 that the TUFW method – using either *Rule−SBD* $\sqrt[4]{K}$ or *Rule−DBD* $\sqrt[4]{K}$ – significantly outperforms the other methods on

on the datasets `a1a`, `a2a`, `a8a`, and `a9a`, for sufficiently small values of ε. For the datasets `w1a`, `w2a`, `w7a`, and `w8a`, the TUFW methods are generally dominated by one of the other metods, with FW being the best on these datasets when ε is smaller. On the datasets `svmguide3`, `phishing`, `ijcnn1`, and `covtype` the TUFW method outperforms the other methods, all the moreso as the required ε gets smaller.

TABLE 4

Comparison of average CPU runtimes (in seconds) required to achieve $\frac{1}{K+1}\sum_{k=0}^{K}\mathcal{G}(x^k) \leq \varepsilon$ *for methods on the non-convex binary classification problem (6.2). (A blank indicates the method used more than 5000 seconds or* $K \geq 10 \times 2^{20}$*.)*

$\mathcal{G}(x^k)$	dataset	n	p	*Rule−SBD* $\sqrt[4]{K}$	*Rule−DBD* $\sqrt[4]{K}$	FW	FW-ada	SPIDER-FW	CASPIDERG	Speed-up
1e-2	a1a	1605	119	6.05	**4.44**	10.46	9.00	15.77		**2.03**
1e-3	a1a	1605	119	90.34	**65.11**	818.40	237.93			**3.65**
1e-4	a1a	1605	119	2225.61	**1453.87**		4953.53			**3.41**
1e-2	a2a	2265	119	6.47	**4.94**	12.62	10.55	13.54		**2.14**
1e-3	a2a	2265	119	93.17	**70.69**	781.05	303.67			**4.30**
1e-4	a2a	2265	119	2168.72	**1389.71**					
1e-2	a8a	22696	123	46.68	41.92	243.81	140.10	**35.70**		0.85
1e-3	a8a	22696	123	668.35	**603.88**		3317.18			**5.49**
1e-4	a8a	22696	123							
1e-2	a9a	32561	123	83.24	79.91	358.08	240.37	**46.05**		0.58
1e-3	a9a	32561	123	1165.35	**1106.16**					
1e-4	a9a	32561	123							
5e-2	w1a	2477	300	8.99	8.82	**0.44**	12.66	0.96	101.83	0.05
1e-2	w1a	2477	300	74.94	102.18	**4.69**	157.90	19.96		0.06
2e-3	w1a	2477	300	1278.96	4063.80	**109.23**	1768.32			0.09
5e-2	w2a	3470	300	12.64	11.90	**0.50**	17.41	1.01	120.51	0.04
1e-2	w2a	3470	300	93.47	122.01	**7.44**	226.70	21.08		0.08
2e-3	w2a	3470	300	900.09	2545.77	**152.15**	2415.59			0.17
5e-2	w7a	24692	300	95.86	90.12	7.79	271.26	**2.16**	595.38	0.02
1e-2	w7a	24692	300	544.57	561.90	80.96	3596.39	**29.75**		0.05
2e-3	w7a	24692	300	4078.15		**1631.47**		2990.71		0.40
5e-2	w8a	49749	300	222.71	216.21	16.41	534.20	**3.25**	1122.68	0.02
1e-2	w8a	49749	300	1292.84	1303.19	176.70		**41.53**		0.03
2e-3	w8a	49749	300			**3268.55**				
1e-1	svmguide3	1243	22	0.47	**0.15**	2.84	1.96			**13.39**
1e-2	svmguide3	1243	22	7.95	**2.41**		74.73			**30.98**
1e-3	svmguide3	1243	22	122.45	**33.68**		2946.23			**87.49**
1e-4	svmguide3	1243	22	2418.83	**804.17**					
1e-1	phishing	11055	68	1.59	1.17	2.36	102.48	**1.17**		0.99
1e-2	phishing	11055	68	17.53	**12.18**	158.49	2506.03			**13.01**
1e-3	phishing	11055	68	216.38	**154.79**					
1e-4	phishing	11055	68	5049.56	**3558.37**					
1e-1	ijcnn1	49990	22	2.32	**0.96**	10.88	103.03	1.58	368.77	**1.64**
1e-2	ijcnn1	49990	22	24.26	**10.00**	728.57	2315.87			**72.85**
1e-3	ijcnn1	49990	22	298.45	**179.40**					
1e-4	ijcnn1	49990	22		**2750.09**					
5e-2	covtype	581012	54	156.41	**110.28**	2152.50	2894.39			**19.52**
1e-2	covtype	581012	54	785.43	**572.97**					
2e-3	covtype	581012	54	4292.90	**3142.24**					

REFERENCES

[1] K. Balasubramanian and S. Ghadimi, *Zeroth-order (non)-convex stochastic optimization via conditional gradient and gradient updates*, in Advances in Neural Information Processing Systems, vol. 31, 2018, pp. 3459–3468.

[2] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, ACM Transactions on Intelligent Systems and Technology (TIST), 2 (2011), p. 27.

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

[4] C. Fang, C. J. Li, Z. Lin, and T. Zhang, *Spider: Near-optimal non-convex optimization via stochastic path integrated differential estimator*, in Advances in Neural Information Processing Systems, vol. 31, 2018, pp. 687–697.

[5] M. Fazel, *Matrix rank minimization with applications*, PhD thesis, PhD thesis, Stanford University, 2002.

[6] M. Frank and P. Wolfe, *An algorithm for quadratic programming*, Naval Research Logistics Quarterly, 3 (1956), pp. 95–110.

[7] R. M. Freund, P. Grigas, and R. Mazumder, *An extended Frank–Wolfe method with "in-face" directions, and its application to low-rank matrix completion*, SIAM Journal on

Optimization, 27 (2017), pp. 319—-346.
[8] H. Gao and H. Huang, *Can stochastic zeroth-order Frank-Wolfe method converge faster for non-convex problems?*, in Proceedings of the 37th International Conference on Machine Learning, vol. 119, PMLR, 2020, pp. 3377–3386.
[9] H. Hassani, A. Karbasi, A. Mokhtari, and Z. Shen, *Stochastic conditional gradient++:(Non) convex minimization and continuous submodular maximization*, SIAM Journal on Optimization, 30 (2020), pp. 3315–3344.
[10] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: Data mining, inference, and prediction*, vol. 2, Springer, 2009.
[11] E. Hazan and H. Luo, *Variance-reduced and projection-free stochastic optimization*, in Proceedings of The 33rd International Conference on Machine Learning, vol. 48, PMLR, 2016, pp. 1263–1271.
[12] F. Huang, L. Tao, and S. Chen, *Accelerated stochastic gradient-free and projection-free methods*, in Proceedings of the 37th International Conference on Machine Learning, vol. 119, PMLR, 2020, pp. 4519–4530.
[13] M. Jaggi, *Revisiting Frank-Wolfe: Projection-free sparse convex optimization*, in Proceedings of the 30th International Conference on Machine Learning, vol. 28, PMLR, 2013, pp. 427–435.
[14] R. Johnson and T. Zhang, *Accelerating stochastic gradient descent using predictive variance reduction*, in Advances in Neural Information Processing Systems, vol. 26, 2013.
[15] S. Lacoste-Julien, *Convergence rate of Frank-Wolfe for non-convex objectives*, arXiv preprint arXiv:1607.00345, (2016).
[16] G. Lan, *The complexity of large-scale convex programming under a linear optimization oracle*, arXiv preprint arXiv:1309.5550, (2013).
[17] G. Lan and Y. Zhou, *Conditional gradient sliding for convex optimization*, SIAM Journal on Optimization, 26 (2016), pp. 1379–1409.
[18] H. Lu and R. M. Freund, *Generalized stochastic Frank-Wolfe algorithm with stochastic "substitute" gradient for structured convex optimization*, Mathematical Programming, 187 (2021), pp. 317–349.
[19] J. Mairal, R. Jenatton, G. Obozinski, and F. Bach, *Convex and network flow optimization for structured sparsity*, Journal of Machine Learning Research, 12 (2011).
[20] S. Mei, Y. Bai, and A. Montanari, *The landscape of empirical risk for nonconvex losses*, The Annals of Statistics, 46 (2018), pp. 2747–2774.
[21] A. Mokhtari, H. Hassani, and A. Karbasi, *Stochastic conditional gradient methods: From convex minimization to submodular maximization*, Journal of Machine Learning Research, (2020).
[22] G. Negiar, G. Dresdner, A. Tsai, L. E. Ghaoui, F. Locatello, R. Freund, and F. Pedregosa, *Stochastic Frank-Wolfe for constrained finite-sum minimization*, in Proceedings of the 37th International Conference on Machine Learning, vol. 119, PMLR, 2020, pp. 7253–7262.
[23] S. J. Reddi, S. Sra, B. Póczos, and A. Smola, *Stochastic Frank-Wolfe methods for nonconvex optimization*, in 2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton), IEEE, 2016, pp. 1244–1251.
[24] A. K. Sahu, M. Zaheer, and S. Kar, *Towards gradient free and projection free stochastic optimization*, in The 22nd International Conference on Artificial Intelligence and Statistics, PMLR, 2019, pp. 3468–3477.
[25] Z. Shen, C. Fang, P. Zhao, J. Huang, and H. Qian, *Complexities in projection-free stochastic non-convex minimization*, in Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics, vol. 89, PMLR, 2019, pp. 2868–2876.
[26] R. Tibshirani, *Regression shrinkage and selection via the lasso*, Journal of the Royal Statistical Society: Series B (Methodological), 58 (1996), pp. 267–288.
[27] V. Vapnik, *Principles of risk minimization for learning theory*, in Advances in Neural Information Processing Systems, vol. 4, 1991, pp. 831–838.
[28] Z. Xiong and R. M. Freund, *Additional results and extensions for the paper "Using Taylor-approximated gradients to improve the Frank-Wolfe method for empirical risk minimization"*, (2022). URL: https://zikaixiong.github.io/papers/fwErmReport.pdf.
[29] A. Yurtsever, S. Sra, and V. Cevher, *Conditional gradient methods via stochastic path-integrated differential estimator*, in Proceedings of the 36th International Conference on Machine Learning, vol. 97, PMLR, 2019, pp. 7282–7291.
[30] M. Zhang, Z. Shen, A. Mokhtari, H. Hassani, and A. Karbasi, *One sample stochastic Frank-Wolfe*, in Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics, vol. 108, PMLR, 2020, pp. 4012–4023.