Federated Frank-Wolfe Algorithm

Ali Dadras(✉), Sourasekhar Banerjee, Karthik Prakhya, and Alp Yurtsever

Umeå University, Umeå, Sweden
{ali.dadras,sourasekhar.banerjee,karthik.prakhya,alp.yurtsever}@umu.se

Abstract. Federated learning (FL) has gained a lot of attention in recent years for building privacy-preserving collaborative learning systems. However, FL algorithms for constrained machine learning problems are still limited, particularly when the projection step is costly. To this end, we propose a Federated Frank-Wolfe Algorithm (FedFW). FedFW features data privacy, low per-iteration cost, and communication of sparse signals. In the deterministic setting, FedFW achieves an ε-suboptimal solution within $\mathcal{O}(\varepsilon^{-2})$ iterations for smooth and convex objectives, and $\mathcal{O}(\varepsilon^{-3})$ iterations for smooth but non-convex objectives. Furthermore, we present a stochastic variant of FedFW and show that it finds a solution within $\mathcal{O}(\varepsilon^{-3})$ iterations in the convex setting. We demonstrate the empirical performance of FedFW on several machine learning tasks.

Keywords: Federated learning · Frank-Wolfe · Conditional gradient method · Projection-free · Distributed optimization

1 Introduction

We present a new variant of the Frank-Wolfe (FW) algorithm, FedFW, designed for the increasingly popular Federated Learning (FL) paradigm in machine learning. Consider the following constrained empirical risk minimization template:

$$\min_{\mathbf{x}\in\mathcal{D}} \quad F(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}), \tag{1}$$

where $\mathcal{D} \subseteq \mathbb{R}^p$ is a convex and compact set. We define the diameter of $\mathcal{D}$ as $D := \max_{\mathbf{x},\mathbf{y}\in\mathcal{D}} \|\mathbf{x}-\mathbf{y}\|$. The function $F : \mathbb{R}^p \to \mathbb{R}$ represents the objective function, and $f_i : \mathbb{R}^p \to \mathbb{R}$ (for $i = 1, \ldots, n$) represent the loss functions of the clients, where n is the number of clients. Throughout, we assume f_i is L-smooth, meaning that it has Lipschitz continuous gradients with parameter L.

FL holds great promise for solving optimization problems over a large network, where clients collaborate under the coordination of a server to find a

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-70352-2_4.

A. Bifet et al. (Eds.): ECML PKDD 2024, LNAI 14943, pp. 58–75, 2024.
https://doi.org/10.1007/978-3-031-70352-2_4

common good model. Privacy is an explicit goal in FL; clients work together towards a common goal by utilizing their own data without sharing it. As a result, FL exhibits remarkable potential for data science applications involving privacy-sensitive information. Its applications range from learning tasks (such as training neural networks) on mobile devices without sharing personal data [1] to medical applications of machine learning, where hospitals collaborate without sharing sensitive patient information [2].

Most FL algorithms focus on unconstrained optimization problems, and extending these algorithms to handle constrained problems typically requires projection steps. However, in many machine learning applications, the projection cost can create a computational bottleneck, preventing us from solving these problems at a large scale. The FW algorithm [3] has emerged as a preferred method for addressing these problems in machine learning. The main workhorse of the FW algorithm is the linear minimization oracle (LMO),

$$\text{lmo}(\mathbf{y}) := \underset{\mathbf{x}\in\mathcal{D}}{\text{argmin}} \ \langle \mathbf{y}, \mathbf{x} \rangle. \tag{2}$$

Evaluating linear minimization is generally less computationally expensive than performing the projection step. A famous example illustrating this is the nuclear-norm constraint: projecting onto a nuclear-norm ball often requires computing a full-spectrum singular value decomposition. In contrast, linear minimization involves finding the top singular vector, a task that can be efficiently approximated using methods such as the power method or Lanczos iterations.

To our knowledge, FW has not yet been explored in the context of FL. This paper aims to close this gap. Our primary contribution lies in adapting the FW method for FL with convergence guarantees.

The paper is organized as follows: Sect. 2 provides a brief review of the literature on FL and the FW method. In Sect. 3, we introduce FedFW. Unlike traditional FL methods, FedFW does not overwrite clients' local models with the global model sent by the server. Instead, it penalizes clients' loss functions by using the global model. We present the convergence guarantees of FedFW in Sect. 3.1. Specifically, our method provably finds a ε-suboptimal solution after $\mathcal{O}(\varepsilon^{-2})$ iterations for smooth and convex objective functions (refer to Theorem 1). In the case of non-convex objectives, the complexity increases to $\mathcal{O}(\varepsilon^{-3})$ (refer to Theorem 2). Section 4 introduces several design variations of FedFW, including a stochastic variant. Section 5 presents numerical experiments on various machine learning tasks with both convex and non-convex objective functions. Finally, Sect. 6 provides concluding remarks along with a discussion on the limitations of the proposed method. Detailed proofs and technical aspects are deferred to the supplementary material.

2 Related Work

Federated Learning. FL is a distributed learning paradigm that, unlike most traditional distributed settings, focuses on a scenario where only a subset of

clients participate in each training round, data is often heterogeneous, and clients can perform different numbers of iterations in each round [4,5]. FedAvg [4] has been a cornerstone in the FL literature, demonstrating practical capabilities in addressing key concerns such as privacy and security, data heterogeneity, and computational costs. Although it is shown that fixed points of some FedAvg variants do not necessarily converge to the minimizer of the objective function, even in the least squares problem [6], and can even diverge [7], the convergence guarantees of FedAvg have been studied under different assumptions (see [8–15] and the references therein). However, all these works on the convergence guarantees of FedAvg are restricted to unconstrained problems.

Constrained or composite optimization problems are ubiquitous in machine learning, often used to impose structural priors such as sparsity or low-rankness. To our knowledge, FedDR [16] and FedDA [17] are the first FL algorithms with convergence guarantees for constrained problems. The former employs Douglas-Rachford splitting, while the latter is based on the dual averaging method [18], to solve composite optimization problems, including constrained problems via indicator functions, within the FL setting. [19] introduced a 'fast' variant of FedDA, achieving rapid convergence rates with linear speedup and reduced communication rounds for composite strongly convex problems. FedADMM [20] was proposed for federated composite optimization problems involving a non-convex smooth term and a convex non-smooth term in the objective. Moreover, [21] proposed a FL algorithm based on a proximal augmented Lagrangian approach to address problems with convex functional constraints. None of these works address our problem template, where the constraints are challenging to project onto but allow for an efficient solution to the linear minimization problem.

Frank-Wolfe Algorithm. The FW algorithm, also known as the conditional gradient method or CGM, was initially introduced in [3] to minimize a convex quadratic objective over a polytope, and was extended to general convex objectives and arbitrary convex and compact sets in [22]. Following the seminal works in [23,24], the method gained popularity in machine learning.

The increasing interest in FW methods for data science applications has led to the development of new results and variants. For example, [25] established convergence guarantees for FW with non-convex objective functions. Additionally, online, stochastic, and variance-reduced variants of FW have been proposed; see [26–31] and the references therein. FW has also been combined with smoothing strategies for non-smooth and composite objectives [32–35], and with augmented Lagrangian methods for problems with affine equality constraints [36,37]. Furthermore, various design variants of FW, such as the away-step and pairwise step strategies, can offer computational advantages. For a comprehensive overview of FW-type methods and their applications, we refer to [38,39].

The most closely related methods to our work are the distributed FW variants. However, the variants in [40–42] are fundamentally different from FedFW as they require sharing gradient information of the clients with the server or with the neighboring nodes. In FedFW, clients do not share gradients, which

is critical for data privacy [43,44]. Other distributed FW variants are proposed in [45–47]. However, the method proposed by [46] is limited to the convex low-rank matrix optimization problem, and the methods in [45,47] assume that the problem domain is block separable.

3 Federated Frank-Wolfe Algorithm

In essence, any first-order optimization algorithm can be adapted for a simplified federated setting by transmitting local gradients to the server at each iteration. These local gradients can be aggregated to compute the full gradient and distributed back to the clients. Although it is possible to implement the standard FW algorithm in FL this way, this baseline has two major problems. First, it relies on communication at each iteration, which raises scalability concerns, as extending this approach to multiple local steps is not feasible. Secondly, sharing raw gradients raises privacy concerns, as sensitive information and data points can be inferred with high precision from transmitted gradients [43]. Consequently, most FL algorithms are designed to exchange local models or step-directions rather than gradients. Unfortunately, a simple combination of the FW algorithm with a model aggregation step fails to find a solution to (1), as we demonstrate with a simple counterexample in the supplementary material. Therefore, developing FedFW requires a special algorithmic approach, which we elaborate on below.

We start by rewriting problem (1) in terms of the matrix decision variable $\mathbf{X} := [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$, as follows:

$$\min_{\mathbf{X} \in \mathcal{D}^n} \frac{1}{n} \sum_{i=1}^{n} f_i(\mathbf{X}\mathbf{e}_i) + \delta_{\mathcal{C}}(\mathbf{X}). \tag{3}$$

Here, $\mathbf{e}_i$ denotes the ith standard unit vector, and $\delta_{\mathcal{C}}$ is the indicator function for the consensus set:

$$\mathcal{C} := \{[\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{p \times n} : \mathbf{x}_1 = \mathbf{x}_2 = \ldots = \mathbf{x}_n\}. \tag{4}$$

It is evident that problems (1) and (3) are equivalent. However, the latter formulation represents the local models of the clients as the columns of the matrix $\mathbf{X}$, offering a more explicit representation for FL.

The original FW algorithm is ill-suited for solving problem (3) due to the non-smooth nature of the objective function because of the indicator function. Drawing inspiration from techniques proposed in [33], we adopt a quadratic penalty strategy to address this challenge. The main idea is to perform FW updates on a surrogate objective which replaces the hard constraint $\delta_{\mathcal{C}}$ with a smooth function that penalizes the distance between $\mathbf{X}$ and the consensus set $\mathcal{C}$:

$$\hat{F}_t(\mathbf{X}) = \frac{1}{n} \sum_{i=1}^{n} f_i(\mathbf{X}\mathbf{e}_i) + \frac{\lambda_t}{2} \mathrm{dist}^2(\mathbf{X}, \mathcal{C}), \tag{5}$$

where $\lambda_t \geq 0$ is the penalty parameter. Note that the surrogate function is parameterized by the iteration counter t, as it is crucial to amplify the impact of the penalty function by gradually increasing λ_t at a specific rate through the iterations. This adjustment will ensure that the generated sequence converges to a solution of the original problem in (3).

To perform an FW update with respect to the surrogate function, first, we need to compute the gradient of $\hat{F}_t$, given by

$$\begin{aligned}\nabla \hat{F}_t(\mathbf{X}) &= \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{X}\mathbf{e}_i)\mathbf{e}_i^\top + \lambda_t(\mathbf{X} - \mathrm{proj}_{\mathcal{C}}(\mathbf{X})) \\ &= \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{x}_i)\mathbf{e}_i^\top + \lambda_t \sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})\mathbf{e}_i^\top\end{aligned} \tag{6}$$

where $\bar{\mathbf{x}} := \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$. Then, we call the linear minimization oracle:

$$\mathbf{S}^t \in \underset{\mathbf{X}\in\mathcal{D}^n}{\operatorname{argmin}} \langle \nabla \hat{F}_t(\mathbf{X}^t), \mathbf{X}\rangle. \tag{7}$$

Since $\mathcal{D}^n$ is separable for the columns of $\mathbf{X}$, we can evaluate (7) in parallel for $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$. Define $\mathbf{s}_i^t$ as

$$\mathbf{s}_i^t \in \underset{\mathbf{x}\in\mathcal{D}}{\operatorname{argmin}} \langle \frac{1}{n}\nabla f_i(\mathbf{x}_i^t) + \lambda_t(\mathbf{x}_i^t - \bar{\mathbf{x}}^t), \mathbf{x}\rangle, \tag{8}$$

where $\bar{\mathbf{x}}^t := \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i^t$. Then, $\mathbf{S}^t = \sum_{i=1}^{n} \mathbf{s}_i^t \mathbf{e}_i^\top$.

Finally, we update the decision variable by $\mathbf{X}^{t+1} = (1-\eta_t)\mathbf{X}^t + \eta_t \mathbf{S}^t$, which can be computed column-wise in parallel:

$$\mathbf{x}_i^{t+1} = (1-\eta_t)\mathbf{x}_i^t + \eta_t \mathbf{s}_i^t, \tag{9}$$

where $\eta_t \in [0, 1]$ is the step-size.

This establishes the fundamental update rule for our proposed algorithm, FedFW. Note that communication is required only during the computation of $\bar{\mathbf{x}}^t$, which constitutes our aggregation step. All other computations can be performed locally by the clients. Algorithm 1 presents FedFW and several design variants, which are further detailed in Sect. 4.

3.1 Convergence Guarantees

This section presents the convergence guarantees of FedFW. We begin with the guarantees for problems with a smooth and convex objective function.

Theorem 1. *Consider problem (1) with L-smooth and convex loss functions f_i. Then, estimation $\bar{\mathbf{x}}^t$ generated by FedFW with step-size $\eta_t = \frac{2}{t+1}$ and penalty parameter $\lambda_t = \lambda_0\sqrt{t+1}$ for any $\lambda_0 > 0$ satisfies*

$$F(\bar{\mathbf{x}}^t) - F(\bar{\mathbf{x}}^*) \leq \mathcal{O}(t^{-1/2}). \tag{10}$$

Algorithm 1. FedFW: Federated Frank-Wolfe Algorithm (+variants)

input $\mathbf{x}_i^1 \in \mathbb{R}^p,\ \forall i \in [n],\ \lambda_t,\ \eta_t,\ \rho_t,\ \bar{\mathbf{x}}^1 = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i^1,\ \mathbf{y}_i^1 = 0,\ \mathbf{d}_i^1 = 0$
for round $t = 1, 2, \ldots, T$ **do**
— **Client**-level local training —
for client $i = 1, 2, \ldots, n$ **do**
— **FedFW:** $\mathbf{g}_i^t = \frac{1}{n}\nabla f_i(\mathbf{x}_i^t) + \lambda_t(\mathbf{x}_i^t - \bar{\mathbf{x}}^t)$
— **FedFW+:** $\mathbf{y}_i^{t+1} = \mathbf{y}_i^t + \lambda_0(\mathbf{x}_i^t - \bar{\mathbf{x}}^t)$
$\mathbf{g}_i^t = \frac{1}{n}\nabla f_i(\mathbf{x}_i^t) + \lambda_t(\mathbf{x}_i^t - \bar{\mathbf{x}}^t) + \mathbf{y}_i^{t+1}$
— **FedFW-sto:** $\mathbf{d}_i^{t+1} = (1-\rho_t)\mathbf{d}_i^t + \rho_t\frac{1}{n}\nabla f_i(\mathbf{x}_i^t, \omega_i^t)$
$\mathbf{g}_i^t = \mathbf{d}_i^{t+1} + \lambda_t(\mathbf{x}_i^t - \bar{\mathbf{x}}^t)$
$\mathbf{s}_i^t = \arg\min\{\langle \mathbf{g}_i^t, \mathbf{x}\rangle : \mathbf{x} \in \mathcal{D}\}$
$\mathbf{x}_i^{t+1} = (1-\eta_t)\mathbf{x}_i^t + \eta_t\mathbf{s}_i^t$
Client communicates $\mathbf{s}_i^t$ to the server.
end for
— **Server**-level aggregation —
$\bar{\mathbf{x}}^{t+1} = (1-\eta_t)\bar{\mathbf{x}}^t + \eta_t\left(\frac{1}{n}\sum_{i=1}^n \mathbf{s}_i^t\right)$
Server communicates $\bar{\mathbf{x}}^{t+1}$ to the clients.
end for

Remark 1. Our proof is inspired by the analysis in [33]. However, a distinction lies in how the guarantees are expressed. In [33], the authors demonstrate the convergence of $\mathbf{x}_i^t$ towards a solution by proving that both the objective residual and the distance to the feasible set converge to zero. In contrast, we establish the convergence of $\bar{\mathbf{x}}^t$, representing a feasible point, focusing only on the objective residual. We present detailed proof in the supplementary material.

It is worth noting that the convergence guarantees of FedFW are slower compared to those of existing unconstrained or projection-based FL algorithms. For instance, in the smooth convex setting with full gradients, FedAvg [4] achieves a rate of $\mathcal{O}(t^{-1})$ in the objective residual. In a convex composite problem setting, FedDA [17] converges at a rate of $\mathcal{O}(t^{-2/3})$. While FedFW guarantees a slower rate of $\mathcal{O}(t^{-1/2})$, it is important to highlight that FedFW employs cheap linear minimization oracles.

Next, we present the convergence guarantees of FedFW for non-convex problems. For unconstrained non-convex problems, the gradient norm is commonly used as a metric to demonstrate convergence to a stationary point. However, this metric is not suitable for constrained problems, as the gradient may not approach zero if the solution resides on the boundary of the feasible set. To address this, we use the following gap function, standard in FW analysis [25]:

$$\text{gap}(\mathbf{x}) := \max_{\mathbf{u}\in\mathcal{D}} \langle \nabla F(\mathbf{x}), \mathbf{x} - \mathbf{u}\rangle. \tag{11}$$

It is straightforward to show that $\text{gap}(\mathbf{x})$ is non-negative for all $\mathbf{x} \in \mathcal{D}$, and it attains zero if and only if $\mathbf{x}$ is a first-order stationary point of Problem (1).

Theorem 2. *Consider problem (1) with L-smooth loss functions f_i. Suppose the sequence $\bar{\mathbf{x}}^t$ is generated by* FedFW *with the fixed step-size $\eta_t = T^{-2/3}$, and penalty parameter $\lambda_t = \lambda_0 T^{1/3}$ for an arbitrary $\lambda_0 > 0$. Then,*

$$\min_{1 \le t \le T} \text{gap}(\bar{\mathbf{x}}^t) \le \mathcal{O}(T^{-1/3}). \tag{12}$$

Remark 2. We present the proof in the supplementary material. Our analysis introduces a novel approach, as [33] does not explore non-convex problems. While our focus is primarily on problems (1) and (3), our methodology can be used to derive guarantees for a broader setting of minimization of a smooth non-convex function subject to affine constraints over a convex and compact set.

As with our previous results, the convergence rate in the non-convex setting is slower compared to FedAvg, which achieves an $\mathcal{O}(t^{-1/2})$ rate in the gradient norm (note the distinction between the gradient norm and squared gradient norm metrics). For composite FL problems with a non-convex smooth loss and a convex non-smooth regularizer, FedDR [16] achieves an $\mathcal{O}(t^{-1/2})$ rate in the norm of a proximal gradient mapping. In contrast, our guarantees are in terms of the Frank-Wolfe (FW) gap. To our knowledge, FedDA does not offer guarantees in the non-convex setting.

3.2 Privacy and Communication Benefits

FedFW offers low communication overhead since the communicated signals are the extreme points of $\mathcal{D}$, which typically have low dimensional representation. For example, if $\mathcal{D}$ is ℓ_1 (resp., nuclear) norm-ball, then the signals $\mathbf{s}_i$ are 1-sparse (resp., rank-one). Additionally, linear minimization is a nonlinear oracle, the reverse operator of which is highly ill-conditioned. Retrieving the gradient from its linear minimization output is generally unfeasible. For example, if $\mathcal{D}$ is the ℓ_1 norm-ball, then $\mathbf{s}_i$ merely reveals the sign of the maximum entry of the gradient. In the case of a box constraint, $\mathbf{s}_i$ only reveals the gradient signs. For the nuclear norm-ball, $\mathbf{s}_i$ unveils only the top eigenvectors of the gradient. Furthermore, FW is robust against additive and multiplicative errors in the linear minimization step [24]; consequently, we can introduce noise to augment data privacy without compromising the convergence guarantees.

In a simple numerical experiment, we demonstrate the privacy benefits of communicating linear minimization outputs instead of gradients. This experiment is based on the Deep Leakage algorithm [43] using the CIFAR100 dataset. Our experiment compares reconstructed images (*i.e.,* leaked data points) obtained from shared gradients versus shared linear minimization outputs, under ℓ_1 and ℓ_2 norm constraints. Figure 1 displays the final reconstructed images alongside the Peak Signal-to-Noise Ratio (PSNR) across iterations. It is evident that reconstruction via linear minimization oracles, particularly under the ℓ_1 ball constraint, is significantly more challenging than raw gradients.

4 Design Variants of FedFW

This section discusses several design variants and extensions of FedFW.

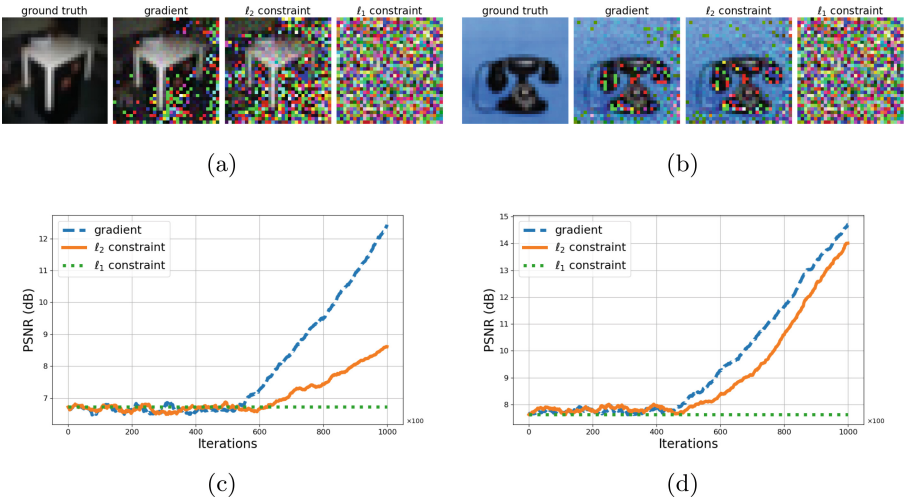


Fig. 1. Privacy benefits of sharing linear minimization outputs vs gradients. The Deep Leakage Algorithm can recover CIFAR-100 data points from shared gradients. Sharing linear minimization outputs enhances privacy. (a) and (b) compares reconstructions from gradients and LMO outputs with ℓ_2 and ℓ_1-norm ball constraints after 10^5 iterations for two different data points. (c) and (d) present the reconstruction PSNR as a function of iterations for the corresponding images.

4.1 FedFW with stochastic gradients

Consider the following stochastic problem template:

$$\min_{\mathbf{x}\in\mathcal{D}} \; F(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{\omega_i}\left[f_i(\mathbf{x},\omega_i)\right]. \tag{13}$$

Here, ω_i is a random variable with an unknown distribution $\mathcal{P}_i$. The client loss function $f_i(\mathbf{x}) := \mathbb{E}_{\omega_i}\left[f_i(\mathbf{x},\omega_i)\right]$ is defined as the expectation over this unknown distribution; hence we cannot compute its gradient. We design FedFW-sto for solving this problem.

We assume that at each iteration, every participating client can independently draw a sample ω_i^t from their distribution $\mathcal{P}_i$. $\nabla f_i(\mathbf{x},\omega_i^t)$ serves as an unbiased estimator of $\nabla f_i(\mathbf{x})$. Additionally, we adopt the standard assumption that the estimator has bounded variance.

Assumption 1 (Bounded variance). *Let $\nabla f_i(\mathbf{x},\omega_i)$ denote the stochastic gradients. We assume that it satisfies the following condition for some $\sigma < \infty$:*

$$\mathbb{E}_{\omega_i}\left[\left\|\nabla f_i(\mathbf{x},\omega_i) - \nabla f_i(\mathbf{x})\right\|^2\right] \le \sigma^2. \tag{14}$$

Unfortunately, FW does not readily extend to stochastic settings by replacing the gradient with an unbiased estimator of bounded variance. Instead, adapting

FW for stochastic settings, in general, requires a variance reduction strategy. Inspired by [30,34], we employ the following averaged gradient estimator to tackle this challenge. We start by $\mathbf{d}_i^0 = \mathbf{0}$, and iteratively update

$$\mathbf{d}_i^{t+1} = (1-\rho_t)\mathbf{d}_i^t + \rho_t \frac{1}{n} \nabla f_i(\mathbf{x}_i^t, \omega_i^t), \tag{15}$$

for some $\rho_t \in (0,1]$. FedFW-sto uses $\mathbf{d}_i^{t+1}$ in place of the gradient in the linear minimization step; pseudocode is shown in Algorithm 1. Although $\mathbf{d}_i^{t+1}$ is not an unbiased estimator, it offers the advantage of reduced variance. The balance between bias and variance can be adjusted by modifying ρ_t, and the analysis relies on finding the right balance, reducing variance sufficiently while maintaining the bias within tolerable limits.

Theorem 3. *Consider problem (13) with L-smooth and convex loss functions f_i. Suppose Assumption 1 holds. Then, the sequence $\bar{\mathbf{x}}^t$ generated by FedFW-sto in Algorithm 1, with step-size $\eta_t = \frac{9}{t+8}$, penalty parameter $\lambda_t = \lambda_0\sqrt{t+8}$ for an arbitrary $\lambda_0 > 0$, and $\rho_t = \frac{4}{(t+7)^{2/3}}$ satisfies*

$$\mathbb{E}[F(\bar{\mathbf{x}}^t)] - F(\mathbf{x}^*) \leq \mathcal{O}(t^{-1/3}). \tag{16}$$

Remark 3. Our analysis in this setting is inspired by [34]; however, we establish the convergence of the feasible point $\bar{\mathbf{x}}^t$. This differs from the guarantees in [34], which demonstrate the convergence of $\mathbf{x}_i^t$ towards a solution by proving that both the expected objective residual and the expected distance to the feasible set converge to zero. We present the detailed proof in the supplementary material.

In the smooth convex stochastic setting, FedAvg achieves a convergence rate of $\mathcal{O}(t^{-1/2})$. This rate also applies to FedDA when addressing composite convex problems. Additionally, under the assumption of strong convexity, Fast-FedDA [19] achieves an accelerated rate of $\mathcal{O}(t^{-1})$. In comparison, FedFW-sto converges with $\mathcal{O}(t^{-1/3})$ rate; however, it benefits from the use of inexpensive linear minimization oracles.

4.2 FedFW with Partial Client Participation

A key challenge in FL is to tackle random device participation schedules. Unlike a classical distributed optimization scheme, in most FL applications, clients have some autonomy and are not entirely controlled by the server. Due to various factors, such as network congestion or resource constraints, clients may intermittently participate in the training process. This obstacle can be tackled in FedFW by employing a block-coordinate Frank-Wolfe approach [48]. Given that the domain of problem (3) is block-separable, we can extend our FedFW analysis to block-coordinate updates.

Suppose that in every round t, the client i participates in the training procedure with a fixed probability of $\mathtt{p}_i \in (0,1]$. For simplicity, we assume the participation rate is the same among all clients, *i.e.*, $\mathtt{p}_1 = \ldots = \mathtt{p}_n := \mathtt{p}$, but

non-uniform participation can be addressed similarly. Instate the convex optimization problem described in Theorem 1 but with the random client participation scheme. At round t, the training procedure follows the same as in Algorithm 1 for the participating clients, and $\mathbf{x}_i^{t+1} = \mathbf{x}_i^t$ for the non-participants. Then, the estimation $\bar{\mathbf{x}}^t$ generated with the step-size $\eta_t = \frac{2}{\mathrm{p}(t-1)+2}$ and penalty parameter $\lambda_t = \lambda_0\sqrt{\mathrm{p}(t-1)+2}$ converges to a solution with rate

$$\mathbb{E}[F(\bar{\mathbf{x}}^t) - F(\mathbf{x}^*)] \leq \mathcal{O}\big((\mathrm{p}\,t)^{-1/2}\big). \tag{17}$$

Similarly, if we consider the non-convex setting of Theorem 2 with randomized client participation, and use the block-coordinate FedFW with step-size $\eta_t = (\mathrm{p}T+1)^{-\frac{2}{3}}$, and penalty parameter $\lambda_t = \lambda_0(\mathrm{p}T+1)^{\frac{1}{3}}$, we get

$$\min_{1\leq t\leq T} \mathbb{E}[\mathrm{gap}(\bar{\mathbf{x}}^t)] \leq \mathcal{O}\big((\mathrm{p}\,T)^{-1/3}\big). \tag{18}$$

The proofs are provided in the supplementary material.

4.3 FedFW with Split Constraints for Stragglers

FL systems are frequently implemented across heterogeneous pools of client hardware, leading to the 'straggler effect'– delays in execution resulting from clients with less computation or communication speeds. In FedFW, we can mitigate this issue by assigning tasks to straggling clients more compatible with their computational capabilities. Theoretically, this adjustment can be achieved through certain special reformulations of the problem defined in (3). Specifically, the constraint $\mathbf{X} \in \mathcal{D}^n$ can be refined to $\mathbf{X} \in \bigcap_{i=1}^n \mathcal{D}_i$, where $\bigcap_{i=1}^n \mathcal{D}_i = \mathcal{D}$. This modification does not affect the solution set, due to the consensus constraint.

In general, in FedFW, most of the computation occurs during the linear minimization step. Suppose that the resources of the client i are limited, particularly for arithmetic computations. In this case, we can select $\mathcal{D}_i$ as a superset of $\mathcal{D}$ where linear minimization computations are more straightforward. For instance, a Euclidean (or Frobenius) norm-ball encompassing $\mathcal{D}$ could be an excellent choice. Then, $\mathbf{s}_i^t$ becomes proportional to the negative of $\mathbf{g}_i^t$ with appropriate normalization based on the radius of $\mathcal{D}_i$, facilitating computation with minimal effort. On the other hand, if the primary bottleneck is communication, we might opt for $\mathcal{D}_i$ characterized by sparse extreme points, such as an ℓ_1-norm ball containing $\mathcal{D}$ or by low-rank extreme points like those in a nuclear-norm ball. This strategy results in sparse (or low-rank) $\mathbf{s}_i^t$, thereby streamlining communication.

4.4 FedFW with Augmented Lagrangian

FedFW employs a quadratic penalty strategy to handle the consensus constraint. We also propose an alternative variant, FedFW+, which is modeled after the augmented Lagrangian strategy in [37]. The pseudocode for FedFW+ is presented in Algorithm 1. We compare the empirical performance of FedFW and FedFW+ in Sect. 5. The theoretical analysis of FedFW+ is omitted here; for further details, we refer readers to [37].

5 Numerical Experiments

In this section, we evaluate and compare the empirical performance of our methods against FedDR, which serves as the baseline algorithm, on the convex multiclass logistic regression (MCLR) problem and the non-convex tasks of training convolutional neural networks (CNN) and deep neural networks (DNN). For each problem, we consider two different choices for the domain $\mathcal{D}$: namely the ℓ_1 and ℓ_2 ball constraints, each with a radius of 10. We assess the models' performance based on validation accuracy, validation loss, and the Frank-Wolfe gap (11). To evaluate the effect of data heterogeneity, we conducted experiments using both IID and non-IID data distributions across clients. The code for the numerical experiments can be accessed via https://github.com/sourasb05/Federated-Frank-Wolfe.git.

Datasets. We use several datasets in our experiments: MNIST [49], CIFAR-10 [50], EMNIST [51], and a synthetic dataset generated as described in [52]. Specifically, the synthetic data is drawn from a multivariate normal distribution, and the labels are computed using softmax functions. We create data points of 60 features and from 10 different labels. For all datasets, we consider both IID and non-IID data distributions across the clients. In the non-IID scenario, each user has data from only 3 labels. We followed this rule for the synthetic data, as well as MNIST, CIFAR10, and EMNIST-10. For EMNIST-62, each user has data from 20 classes, with unequal distribution among users.

5.1 Comparison of Algorithms in the Convex Setting

We tested the performance of the algorithms on the strongly convex MCLR problem using the MNIST and CIFAR-10 datasets as well as the synthetic dataset. Table 1 presents the test accuracy results for the algorithms with IID and non-IID data distributions, and for two different choices of $\mathcal{D}$. In these experiments, we simulated FL with 10 clients, all participating fully ($\mathtt{p} = 1$). We ran the algorithms for 100 communication rounds, with one local iteration per round.

5.2 Comparison of Algorithms in the Non-convex Setting

For the experiments in the non-convex setting we trained CNNs using the MNIST dataset and a DNN with two hidden layers using the synthetic dataset. We considered an FL system with 10 clients and full participation ($\mathtt{p} = 1$). Similar to the previous case, we evaluated IID and non-IID data distributions as well as different choices of $\mathcal{D}$, and ran the methods for 100 communication rounds with a single local training step. Table 2 summarizes the resulting test accuracies.

Table 1. Comparison of algorithms on the convex MCLR problem with different datasets and choices of $\mathcal{D}$. We consider both IID and non-IID data distributions. The numbers represent test accuracy.

	IID			non-IID		
	MNIST	Synthetic	CIFAR10	MNIST	Synthetic	CIFAR10
	ℓ_2 constraint					
FedDR	**89.59**(±**0.0003**)	78.24(±0.007)	**39.95**(±**0.001**)	83.72(±0.001)	92.97(±0.005)	37.79(±0.004)
FedFW	86.96(±0.009)	**80.20**(±**0.01**)	36.30(±0.001)	86.95(±0.001)	**94.81**(±**0.001**)	**38.13**(±**0.003**)
FedFW+	86.50(±0.001)	79.96(±0.001)	36.30(±0.001)	**86.98**(±**0.001**)	94.56(±0.009)	37.20(±0.004)
	ℓ_1 constraint					
FedDR	72.18(±0.0004)	79.00(±0.004)	**23.25**(±**0.00**)	74.29(±0.0)	**93.81**(±**0.009**)	24.77(±0.0)
FedFW	**78.07**(±**0.005**)	81.63(±0.01)	21.86(±0.003)	**80.54**(±**0.0**)	90.84(±0.003)	25.08(±0.004)
FedFW+	69.17(±0.004)	**81.92**(±**0.008**)	21.99(0.006)	71.32(±0.002)	91.20(±0.006)	**25.16**(±**0.008**)

Table 2. Comparison of algorithms on the non-convex tasks. We train a CNN using MNIST, and a DNN with synthetic data. We consider IID and non-IID data distributions, and different choices of $\mathcal{D}$. The numbers show test accuracy.

	IID		non-IID	
	MNIST	Synthetic	MNIST	Synthetic
	ℓ_2 constraint			
FedDR	**96.89**(±**0.0009**)	75.96(±0.03)	88.93(±0.013)	93.85(±0.009)
FedFW	95.87(±0.01)	81.70(±0.008)	**92.70**(±**0.002**)	**96.13**(±**0.007**)
FedFW+	95.05(±0.005)	**81.96**(±**0.006**)	91.79(±0.005)	96.08(±0.004)
	ℓ_1 constraint			
FedDR	11.72(±0.0)	**78.59**(±**0.006**)	16.75(±0.01)	**93.51**(±**0.006**)
FedFW	**23.88**(±**0.005**)	75.52(±0.008)	**37.62**(±**0.008**)	91.53(±0.01)
FedFW+	20.40(±0.003)	76.44(±0.004)	36.27(±0.006)	91.96(±0.003)

5.3 Comparison of Algorithms in the Stochastic Setting

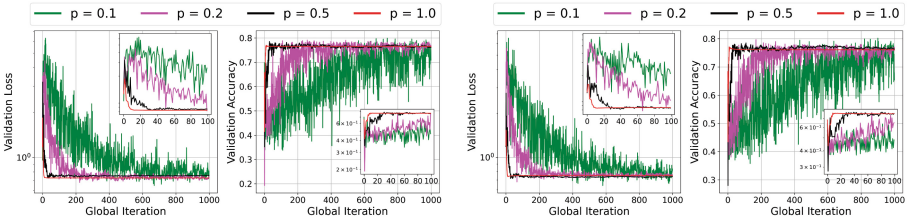
Finally, we compared the performance of FedFW-sto against FedDR in the stochastic setting, where only stochastic gradients are accessible. For this experiment, we consider an FL network with 100 clients with full participation ($\mathtt{p} = 1$). Over this network, we trained the MCLR model using EMNIST-10, EMNIST-62, CIFAR10, and the synthetic dataset. We used a mini-batch size of 64, one local iteration per communication round, and ran the algorithms for 300 communication rounds. Table 3 summarizes the test accuracies obtained in this experiment. FedFW-sto outperformed FedDR in our experiments in the stochastic setting.

5.4 Impact of Hyperparameters

We conclude our experiments with an ablation study to investigate how varying hyperparameters impact the performance of FedFW.

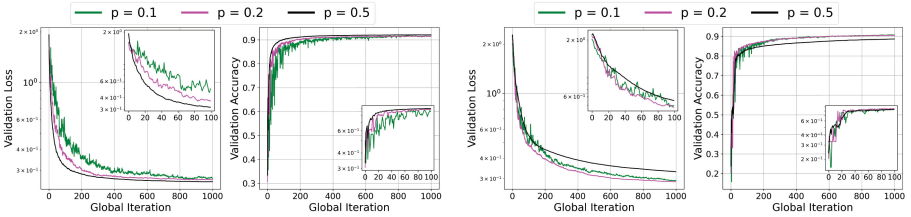
Table 3. Comparison of algorithms in the stochastic setting on the convex MCLR problem with different datasets and ℓ_2 ball constraint. We consider both IID and non-IID data distributions. The numbers represent test accuracy.

	IID		non-IID	
	EMNIST-10	EMNIST-62	EMNIST-10	EMNIST-62
FedDR	92.18(±0.01)	39.58(±0.00)	85.70(±0.002)	41.09(±0.002)
FedFW-sto	**93.79**(±**0.00**)	**41.22**(±**0.00**)	**91.88**(±**0.01**)	**60.51**(±**0.002**)
	Synthetic	CIFAR10	Synthetic	CIFAR10
FedDR	67.68(±0.003)	36.39(±0.004)	84.70(±0.01)	34.91(±0.008)
FedFW-sto	**72.01**(±**0.004**)	**38.52**(±**0.01**)	**87.32**(±**0.003**)	**37.83**(±**0.01**)



(a) $\lambda_0 = 0.01$ (b) $\lambda_0 = 0.001$

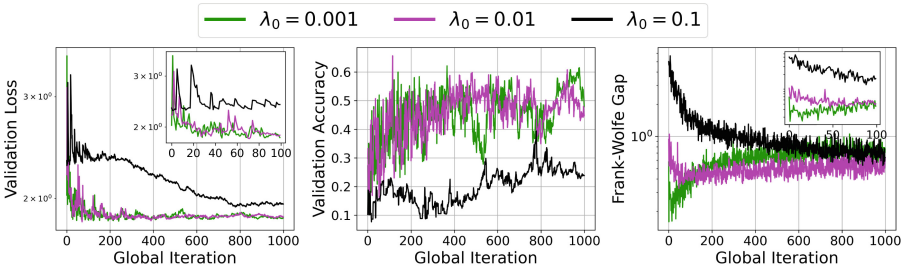
Fig. 2. Effect of participation p on FedFW. The experiment was conducted with MCLR using synthetic data, an ℓ_1 constraint, and two different choices of λ_0.



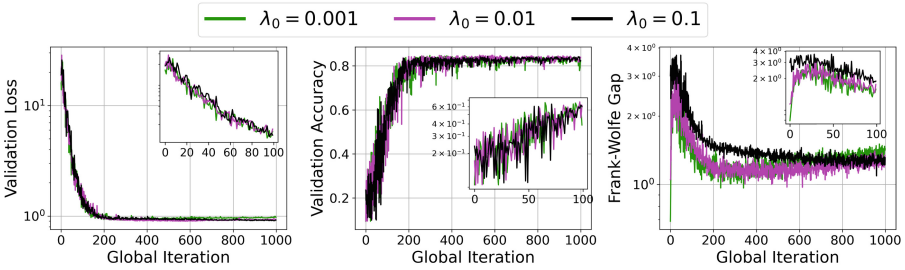
(a) FedFW (b) FedFW+

Fig. 3. Effect of participation p on FedFW and FedFW+. We trained a DNN model using synthetic data, an ℓ_2 constraint, and a fixed $\lambda_t = 10^{-3}$.

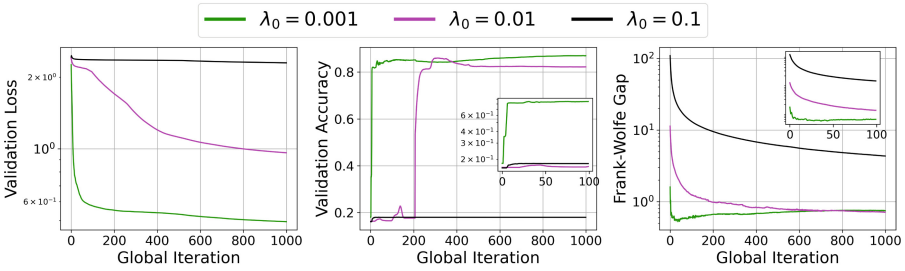
Impact of Partial Participation (p). Figure 2 shows the validation accuracy and loss of FedFW algorithm for synthetic data and MCLR model. Figure 3 depicts the validation accuracy and loss of FedFW and FedFW+ algorithms for synthetic data and DNN model. Both convex and non-convex experiments show faster convergence for higher participation probability. It is worth mentioning that variations in λ_0 do not alter the influence of p. These observations are in accordance with the theoretical guarantees presented in Sect. 4.2.



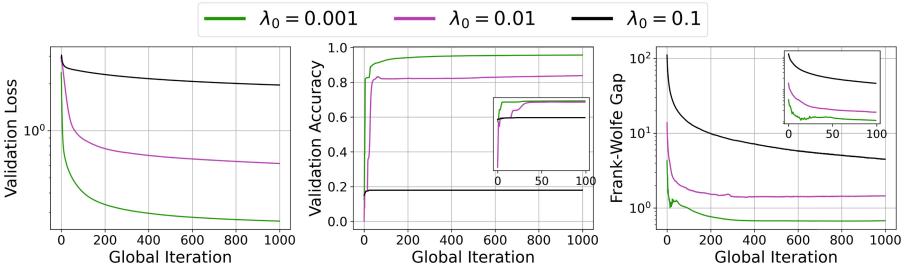
(a) Convex MCLR with MNIST data, participation ratio $\mathtt{p} = 0.5$, and ℓ_1 constraint.



(b) Convex MCLR with MNIST data, participation ratio $\mathtt{p} = 0.5$, and ℓ_2 constraint.



(c) Non-convex DNN with synthetic data, full participation $\mathtt{p} = 1$, and ℓ_1 constraint.



(d) Non-convex DNN with synthetic data, full participation $\mathtt{p} = 1$, and ℓ_2 constraint.

Fig. 4. Effect of the initial penalty (λ_0) on FedFW. (a) and (b) show the results for the convex setting, (c) and (d) demonstrates the non-convex setting.

Impact of Initial Penalty Parameter (λ_0). Figure 4 illustrates the effect of hyperparameters on the convergence of loss, Frank-Wolfe gap, and validation accuracy of the algorithms. A higher λ_0 leads to a larger gap in the initial iterations of the algorithm due to its regularization effect. In other words, increasing λ_0 enforces the update direction towards the consensus set, which in turn increases the gap value in the first iteration. The exact expressions for the constants in the convergence guarantees, which are detailed in the supplementary material, can guide the optimal choice of λ_0.

6 Conclusions

We introduced a FW-type method for FL and established its theoretical convergence rates. The proposed method, FedFW, guarantees $\mathcal{O}(t^{-1/2})$ convergence rates when the objective function smooth and convex. If we remove the convexity assumption, the rate reduces to $\mathcal{O}(t^{-1/3})$. With access to only stochastic gradients, FedFW achieves an $\mathcal{O}(t^{-1/3})$ convergence rate in the convex setting. Additionally, we proposed an empirically faster version of FedFW by incorporating an augmented Lagrangian dual update.

We conclude with a brief discussion on the limitations of our work. The primary limitation of FedFW is its slower convergence rates compared to state-of-the-art FL methods. Developing a tighter bound for FedFW, with multiple local steps, is an area for future research. Additionally, the analysis of FedFW+ is left to future work. Another important piece of future work is the convergence analysis of FedFW-sto for non-convex objectives. Finally, the development and analysis of an extension for asynchronous updates also remain as future work.

Acknowledgements. This work was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. We also acknowledge support from the Swedish Research Council under the grant registration number 2023-05476. The computations were enabled by the Berzelius resource provided by the Knut and Alice Wallenberg Foundation at the National Supercomputer Centre. Additionally, computations in an earlier version of this work were enabled by resources provided by the Swedish National Infrastructure for Computing (SNIC) at Chalmers Centre for Computational Science and Engineering (C3SE) partially funded by the Swedish Research Council through grant agreement no. 2018-05973. We appreciate the discussions with Yikun Hou on the numerical experiments and implementation. We acknowledge the use of OpenAI's ChatGPT for editorial assistance in preparing this manuscript.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Lim, W.Y.B., et al.: Federated learning in mobile edge networks: a comprehensive survey. IEEE Commun. Surv. Tutor. **22**(3), 2031–2063 (2020)
2. Wang, J., et al.: A field guide to federated optimization. arXiv:2107.06917 (2021)
3. Frank, M., Wolfe, P.: An algorithm for quadratic programming. Naval Res. Logist. Q. **3**(1–2), 95–110 (1956)
4. McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282. PMLR (2017)
5. Konečnỳ, J., McMahan, H.B., Ramage, D., Richtárik, P.: Federated optimization: distributed machine learning for on-device intelligence. arXiv:1610.02527 (2016)
6. Pathak, R., Wainwright, M.J.: Fedsplit: an algorithmic framework for fast federated optimization. In: Advances in Neural Information Processing Systems, vol. 33, pp. 7057–7066 (2020)
7. Zhang, X., Hong, M., Dhople, S., Yin, W., Liu, Y.: Fedpd: a federated learning framework with adaptivity to non-IID data. IEEE Trans. Sig. Process. **69**, 6055–6070 (2021)
8. Stich, S.U.: Local SGD converges fast and communicates little. arXiv:1805.09767 (2018)
9. Li, X., Huang, K., Yang, W., Wang, S., Zhang, Z.: On the convergence of FedAVG on non-IID data. In: International Conference on Learning Representations (2019)
10. Haddadpour, F., Kamani, M.M., Mahdavi, M., Cadambe, V.: Local SGD with periodic averaging: tighter analysis and adaptive synchronization. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
11. Yu, H., Yang, S., Zhu, S.: Parallel restarted SGD with faster convergence and less communication: demystifying why model averaging works for deep learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, no. 1, pp. 5693–5700 (2019)
12. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. Proc. Mach. Learn. Syst. **2**, 429–450 (2020)
13. Woodworth, B., et al.: Is local SGD better than minibatch SGD? In: International Conference on Machine Learning, pp. 10334–10343. PMLR (2020)
14. Woodworth, B.E., Patel, K.K., Srebro, N.: Minibatch vs local SGD for heterogeneous distributed learning. In: Advances in Neural Information Processing Systems, vol. 33, pp. 6281–6292 (2020)
15. Al-Shedivat, M., Gillenwater, J., Xing, E., Rostamizadeh, A.: Federated learning via posterior averaging: a new perspective and practical algorithms. arXiv:2010.05273 (2020)
16. Tran Dinh, Q., Pham, N.H., Phan, D., Nguyen, L.: FedDR-randomized Douglas-Rachford splitting algorithms for nonconvex federated composite optimization. In: Advances in Neural Information Processing Systems, vol. 34, pp. 30326–30338 (2021)
17. Yuan, H., Zaheer, M., Reddi, S.: Federated composite optimization. In: International Conference on Machine Learning, pp. 12253–12266. PMLR (2021)
18. Nesterov, Y.: Primal-dual subgradient methods for convex problems. Math. Program. **120**(1), 221–259 (2009)
19. Bao, Y., Crawshaw, M., Luo, S., Liu, M.: Fast composite optimization and statistical recovery in federated learning. In: International Conference on Machine Learning, pp. 1508–1536. PMLR (2022)

20. Wang, H., Marella, S., Anderson, J.: Fedadmm: a federated primal-dual algorithm allowing partial participation. In: 2022 IEEE 61st Conference on Decision and Control (CDC), pp. 287–294. IEEE (2022)
21. He, C., Peng, L., Sun, J.: Federated learning with convex global and local constraints. In: OPT 2023: Optimization for Machine Learning (2023)
22. Levitin, E.S., Polyak, B.T.: Constrained minimization methods. USSR Comput. Math. Math. Phys. **6**(5), 1–50 (1966)
23. Hazan, E.: Sparse approximate solutions to semidefinite programs. In: Laber, E.S., Bornstein, C., Nogueira, L.T., Faria, L. (eds.) LATIN 2008. LNCS, vol. 4957, pp. 306–316. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78773-0_27
24. Jaggi, M.: Revisiting frank-wolfe: projection-free sparse convex optimization. In: International Conference on Machine Learning, pp. 427–435. PMLR (2013)
25. Lacoste-Julien, S.: Convergence rate of Frank-Wolfe for non-convex objectives. arXiv:1607.00345 (2016)
26. Hazan, E., Kale, S.: Projection-free online learning. In: International Conference on Machine Learning. PMLR (2012)
27. Hazan, E., Luo, H.: Variance-reduced and projection-free stochastic optimization. In: International Conference on Machine Learning, pp. 1263–1271. PMLR (2016)
28. Reddi, S.J., Sra, S., Póczos, B., Smola, A.: Stochastic Frank-Wolfe methods for nonconvex optimization. In: 2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp. 1244–1251. IEEE (2016)
29. Yurtsever, A., Sra, S., Cevher, V.: Conditional gradient methods via stochastic path-integrated differential estimator. In: International Conference on Machine Learning. pp. 7282–7291. PMLR (2019)
30. Mokhtari, A., Hassani, H., Karbasi, A.: Stochastic conditional gradient methods: from convex minimization to submodular maximization. J. Mach. Learn. Res. **21**(105), 1–49 (2020)
31. Négiar, G., et al.: Stochastic Frank-Wolfe for constrained finite-sum minimization. In: international Conference on Machine Learning, pp. 7253–7262. PMLR (2020)
32. Lan, G.: An optimal method for stochastic composite optimization. Math. Program. **133**(1), 365–397 (2012)
33. Yurtsever, A., Fercoq, O., Locatello, F., Cevher, V.: A conditional gradient framework for composite convex minimization with applications to semidefinite programming. In: International Conference on Machine Learning, pp. 5727–5736. PMLR (2018)
34. Locatello, F., Yurtsever, A., Fercoq, O., Cevher, V.: Stochastic Frank-Wolfe for composite convex minimization. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
35. Dresdner, G., Vladarean, M.L., Rätsch, G., Locatello, F., Cevher, V., Yurtsever, A.: Faster one-sample stochastic conditional gradient method for composite convex minimization. In: International Conference on Artificial Intelligence and Statistics, pp. 8439–8457. PMLR (2022)
36. Gidel, G., Pedregosa, F., Lacoste-Julien, S.: Frank-Wolfe splitting via augmented Lagrangian method. In: International Conference on Artificial Intelligence and Statistics, pp. 1456–1465. PMLR (2018)
37. Yurtsever, A., Fercoq, O., Cevher, V.: A conditional-gradient-based augmented Lagrangian framework. In: International Conference on Machine Learning, pp. 7272–7281. PMLR (2019)
38. Kerdreux, T.: Accelerating conditional gradient methods. Ph.D. thesis, Université Paris sciences et lettres (2020)

39. Bomze, I.M., Rinaldi, F., Zeffiro, D.: Frank–Wolfe and friends: a journey into projection-free first-order optimization methods. 4OR **19**, 313–345 (2021)
40. Wai, H.T., Lafond, J., Scaglione, A., Moulines, E.: Decentralized Frank-Wolfe algorithm for convex and nonconvex problems. IEEE Trans. Autom. Control **62**(11), 5522–5537 (2017)
41. Mokhtari, A., Hassani, H., Karbasi, A.: Decentralized submodular maximization: bridging discrete and continuous settings. In: International Conference on Machine Learning, pp. 3616–3625. PMLR (2018)
42. Gao, H., Xu, H., Vucetic, S.: Sample efficient decentralized stochastic Frank-Wolfe methods for continuous DR-submodular maximization. In: Thirtieth International Joint Conference on Artificial Intelligence (2021)
43. Zhu, L., Liu, Z., Han, S.: Deep leakage from gradients. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
44. Li, Z., Zhang, J., Liu, L., Liu, J.: Auditing privacy defenses in federated learning via generative gradient leakage. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10132–10142 (2022)
45. Wang, Y.X., Sadhanala, V., Dai, W., Neiswanger, W., Sra, S., Xing, E.: Parallel and distributed block-coordinate Frank-Wolfe algorithms. In: International Conference on Machine Learning, pp. 1548–1557. PMLR (2016)
46. Zheng, W., Bellet, A., Gallinari, P.: A distributed Frank-Wolfe framework for learning low-rank matrices with the trace norm. Mach. Learn. **107**(8), 1457–1475 (2018)
47. Zhang, M., Zhou, Y., Ge, Q., Zheng, R., Wu, Q.: Decentralized randomized block-coordinate Frank-Wolfe algorithms for submodular maximization over networks. IEEE Trans. Syst. Man Cybern. Syst. (2021)
48. Lacoste-Julien, S., Jaggi, M., Schmidt, M., Pletscher, P.: Block-coordinate Frank-Wolfe optimization for structural SVMs. In: International Conference on Machine Learning, pp. 53–61. PMLR (2013)
49. LeCun, Y., et al.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
50. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images, Toronto, ON, Canada (2009)
51. Cohen, G., et al.: EMNIST: extending MNIST to handwritten letters. In: IJCNN, pp. 2921–2926. IEEE (2017)
52. Dinh, T., Tran, C., Nguyen, N., Personalized federated learning with Moreau envelopes. J. Adv. Neural Inf. Process. Syst. **33**, 21394–21405 (2020)