An Uplink Communication-Efficient Approach to Featurewise Distributed Sparse Optimization With Differential Privacy

Jian Lou and Yiu-ming Cheung, *Fellow, IEEE*

***Abstract*— In sparse empirical risk minimization (ERM) models, when sensitive personal data are used, e.g., genetic, healthcare, and financial data, it is crucial to preserve the differential privacy (DP) in training. In many applications, the information (i.e., features) of an individual is held by different organizations, which give rise to the prevalent yet challenging setting of the featurewise distributed multiparty model training. Such a setting is also beneficial to the scalability when the number of features exceeds the computation and storage capacity of a single node. However, existing private sparse optimizations are limited to centralized and samplewise distributed datasets only. In this article, we develop a differentially private algorithm for the sparse ERM model training under the featurewise distributed datasets setting. Our algorithm comes with guaranteed DP, nearly optimal utility, and reduced uplink communication complexity. Accordingly, we present a more generalized convergence analysis for block-coordinate Frank–Wolfe (BCFW) under *arbitrary sampling* (denoted as BCFW-AS in short), which significantly extends the known convergence results that apply to two specific sampling distributions only. To further reduce the uplink communication cost, we design an active private feature sharing scheme, which is new in both design and analysis of BCFW, to guarantee the convergence of communicating Johnson–Lindenstrauss transformed features. Empirical studies justify the new convergence as well as the nearly optimal utility theoretical results.**

***Index Terms*— Differential privacy (DP), distributed optimization, empirical risk minimization (ERM), sparse optimization.**

I. Introduction

SPARSE empirical risk minimization (ERM) [1]–[4] is an important machine learning model, which learns from data collected from individuals. Despite its usefulness and the large body of research for improving its efficiency [5]–[7], its involvement of sensitive individual data poses increasing privacy concerns, especially in applications of healthcare, financial, and biomedicine. To avoid breaching the privacy of the individuals, privacy-preserving techniques have been developed to ensure that the adversary cannot infer any individual data from the output of the learning process. Beginning with the seminal work [8], which proposes to carry out the private ERM training under the formal statistical differential privacy (DP) notion [9], different types of differentially private optimization algorithms have been developed to suit various computing contexts. In particular, existing works focus on training the model with centralized datasets [10]–[14] and samplewise distributed datasets [15]–[18]. For example, when samples distribute among user sites, Lou *et al.* [16] and Han *et al.* [18] proposed privacy-preserving strategies for the stochastic (sub) gradient (SGD) method to avoid sensitive user information leakage during distributed optimization. Agarwal *et al.* [15] and Jin *et al.* [17] further reduced the uplink communication cost by utilizing gradient quantization techniques [19]–[24]. These existing research studies, although providing a decent privacy-preserving guarantee with optimal utility and efficiency for centralized and samplewise distributed settings, leave the featurewise distributed private training barely studied.

However, such a featurewise distributed setting appears in many real applications, where the information describing an individual is collected and held by different parties such as different sets of sensory systems, different organizations, different hospitals, and so on. For example, in the application domain of healthcare and biomedicine [25]–[27], a person's medical records and biometric information (like genomic data) are sensitive personal information that can be held by several medical organizations and clinics. Besides, for applications whose datasets have a large number of features, splitting features among many computing nodes help improve the scalability of the system. Unfortunately, the existing differentially private ERM algorithms are limited to centralized and samplewise distributed settings only.

Under the circumstances, this article will concentrate on the featurewise distributed dataset setting. We begin by identifying two key design considerations of such a distributed algorithm: uplink communication cost, and utility. Considering a typical "one server and many users" distributed training architecture, we call the information from user nodes to the server node as uplink communication, while calling the information broadcasted by the server node to all users by downlink communication. The uplink communication is com-

Manuscript received September 13, 2019; revised March 14, 2020, June 23, 2020, and August 9, 2020; accepted August 29, 2020. Date of publication September 17, 2020; date of current version October 6, 2021. This work was supported in part by the NSFC under Grant 61672444, in part by HKBU under Grant RC-FNRA-IG/18-19/SCI/03 and Grant RC-IRCMs/18-19/SCI/01, and in part by the ITF of ITC of the Government of the Hong Kong SAR under Project ITS/339/18. *(Corresponding author: Yiu-ming Cheung.)*

Jian Lou is with the Department of Computer Science, Emory University, Atlanta, GA 30322 USA (e-mail: jian.lou@emory.edu).

Yiu-ming Cheung is with the Department of Computer Science, Hong Kong Baptist University, Hong Kong (e-mail: ymc@comp.hkbu.edu.hk).

This article has supplementary downloadable material available at https://ieeexplore.ieee.org, provided by the authors.

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2020.3020955

puted based on the sensitive individual information held by the user nodes, which can release individual privacy. On the other hand, downlink communication is considered public because it will maintain DP property even without any privacy protection mechanism. It is from the postprocessing property of the DP [9], the postprocessed information (i.e., downlink communication here) will not further compromise the privacy of the algorithm, as long as the uplink communication is private enough. As such, the key design consideration for private training using a distributed dataset lies in preventing the adversary from inferring sensitive individual information by spying on the uplink communication. Intuitively, minimizing the uplink communication complexity means less exposure of sensitive data and reduced potential of personal data leakage. Thus, less uplink communication would generally require less privacy protection budget for ensuring DP. The second is the utility of the algorithm. In general, with the privacy-preserving restriction, the utility of a privatized algorithm tends to drop. In order to avoid deteriorating the utility of the algorithm, it requires careful balancing the privacy and utility.

A. Motivating Example: Federated Learning (FL)

A motivating example of the above setting is FL [28]–[30], which is an emerging paradigm of the more general distributed learning that receives increasing research interest nowadays. The FL aims to train a joint global model by the collective participation of all clients (e.g., model devices or organizations) under the coordination of a central server, while requiring no communication of the local raw data but only intermediate model parameters. The FL is recognized as a superior specification of the general distributed learning in terms of the privacy-preserving due to the elimination of the raw data transmission. In addition, an ideal FL algorithm should have low communication cost since the participating clients in general have limited communication capacity. Thus, "privacy and communication efficiency are always first-order concerns in FL," as quoted from [30].

As can be seen, our aims of jointly training an accurate model, while reducing the uplink communication cost and ensuring privacy-preserving, are well-aligned with the FL, the cross-silo FL [30] to be specific. However, most existing FL studies (and even the broader distributed learning) focus on the samplewise distributed setting, while the featurewise distributed setting has not been properly addressed in existing research. Thus, this article can be regarded as studying an under-studied FL setting that focuses on the featurewise distributed setting.

B. Contributions

We propose a distributed private block-coordinate Frank–Wolfe under arbitrary sampling (BCFW-AS) algorithm for solving sparse ERM specific to the featurewise distributed computing context. Moreover, the proposed algorithm features uplink communication efficiency and the same nearly optimal utility as that in the centralized setting. At first glance, it seems straightforward by integrating the report-noisy-max for privacy protection [31] with the existing distributed FW design. However, such direct combinations either: 1) do not have known differentially private strategy for communicating active features, which is indispensable for computing local partial gradient (PG) under the featurewise distributed setting [32] or 2) require each user node to have full replication of the entire feature set, which is undesirable as local features need to be communicated with an extra preprocessing step [33]. The seemingly different aspects of incapabilities actually attribute to the same reason: existing randomized BCFW algorithms have limited convergence guarantees that apply to two simple block sampling distributions only, i.e., uniform serial sampling and τ-nice sampling. In this regard, we make primarily two contributions.

1) BCFW-AS : After revisiting existing BCFW algorithms and analysis, we develop a more general convergence analysis for BCFW-AS. In addition to a universal convergence analysis, our approach enjoys greater flexibility than existing analysis [33], [34] whose analysis is highly dependent on the specific samplings. Furthermore, in contrast to existing expected separable overapproximation (ESO) inequality-based coordinate descent under arbitrary sampling algorithms [35]–[38], we develop our convergence analysis by introducing a new notion called expected curvature, which is a more fundamental quantity for FW algorithms than ESO-based counterparts.

2) Private BCFW for Featurewise Distributed Dataset: We provide an uplink communication efficient private algorithm based on the inexact BCFW-AS for solving the sparse ERM. We design two key building components: private index computation and private active feature sharing. Both are able to reduce uplink communication cost and provide privacy protection. Our analysis shows the algorithm has favorable properties like strict DP, nearly optimal utility, reduced uplink communication cost.

Our previous conference paper [39] only provides preliminary theoretical results, and the private algorithm is only applicable to LASSO (i.e., with least square loss) problem without any empirical evidence. In this extended paper, we provide a more thorough and detailed theoretical analysis and comparison for the BCFW-AS algorithm, including both exact and inexact BCFW-AS algorithms. Furthermore, we develop a more general differentially private algorithm whose application is not limited to LASSO. For all the major theoretical results, rigorous proofs are derived. In addition to theoretical analysis, this extended version provides empirical evidence to verify all major theoretical results one by one, and also studies the impact of the parameters to the algorithms, which are not provided in the conference version.

The remainder of this article is organized as follows. Section II introduces the related work. Section III provides the BCFW-AS algorithm and its convergence analysis. Section IV proposes the algorithm for private feature-distributed ERM training, along with guaranteed DP, nearly optimal utility, and reduced uplink communication complexity. In Section V, we present the empirical evidence to support the theoretical guarantees. Section VI gives the concluding remarks. Finally, the Supplementary Material contains additional lemmas and proofs.

TABLE I
SYMBOLS AND NOTATIONS USED IN THIS ARTICLE

Symbol	Definition
$\boldsymbol{a}, \boldsymbol{A}, \boldsymbol{A}^\top$	Vector, Matrix, Matrix transpose
$\boldsymbol{e}_i$	Standard basis vector
$[d]$	Index set {1,2,....,d}
$\mathcal{T}$	Index subset, i.e. $\mathcal{T} \subseteq [d]$
$f(\boldsymbol{x})$	Loss function $f(\cdot)$ with decision variable $\boldsymbol{x}$
$\mathcal{M}$	Constraint set for $\boldsymbol{x}$, i.e. $\boldsymbol{x} \in \mathcal{M}$
$\boldsymbol{x}_{(\mathcal{T})}, \nabla_{(\mathcal{T})} f(\boldsymbol{x})$	partial vector and partial gradient on $\mathcal{T}$
$\mathbb{D}$	all data for training $\mathbb{D} : n \times d$
$\|\cdot\|_1, \|\cdot\|_F$	ℓ_1-norm, Frobenius norm
$\|\cdot\|_{(\mathcal{T})}, \|\cdot\|^*_{(\mathcal{T})}$	Partial primal/dual norm computed on $\mathcal{T}$
$\circ$	Hadamard (element-wise) multiplication

II. RELATED WORK

Table I summarizes frequently used notation in this article. In this section, we first make an overview of known convergence guarantees for BCFW algorithms in Section II-A. Next, Sections II-B and II-C introduce the basics of the DP and the existing differentially private methods for training under featurewise distributed setting, correspondingly. Then, Section II-D introduces the state-of-the-art communication-efficient distributed optimization approaches.

A. Sampling Distributions and Known Convergence Results for BCFW Algorithms

We first revisit the BCFW algorithms by interpreting them as *randomized* BCFW algorithms under the corresponding sampling distributions to highlight the association of the convergence analysis with the types of sampling distributions. Table III summarizes the previous and our new BCFW algorithms, samplings, and convergence guarantees. We follow the naming convention used in existing arbitrarily sampled coordinate descent papers [35], [37], [38] for referring several common samplings.

1) Elementary Sampling: Jaggi [40] proposes the conventional deterministic FW algorithm using the full gradient per-iteration, which can be seen as sampling the coordinates under elementary sampling with set $[d]$, i.e., sampling set $[d]$ with probability one. With the step-size $\gamma^t = (2/(t+2))$, it guarantees $h^t = f(\boldsymbol{x}^t) - f(\boldsymbol{x}^*) \leq (2C_f/(t+2))$, where $x^* \in \mathcal{M}$ denotes an optimum and C_f is the curvature of $f(x)$ on the whole constraint set $\mathcal{M}$, which measures the nonlinearity of $f(\boldsymbol{x})$ on the entire constraint set $\mathcal{M}$, reflecting the geometric property of $f(\boldsymbol{x})$ on $\mathcal{M}$

$$C_f := \sup_{\substack{\boldsymbol{x},\boldsymbol{s}\in\mathcal{M},\\ \gamma\in[0,1]}} \frac{2}{\gamma^2}\Big(f(\boldsymbol{x}+\gamma(\boldsymbol{s}-\boldsymbol{x})) - f(\boldsymbol{x}) - \gamma\langle \boldsymbol{s}-\boldsymbol{x}, \nabla f(\boldsymbol{x})\rangle\Big). \tag{1}$$

The algorithm dFW [32] is a distributed FW method. During one communication pass, each worker evaluates the partial linear oracle (LO) based on the local features and then sends both the partial LO index and the associated local duality gap value to the sever node for comparison. Subsequently, the partial LO with the maximum local duality gap is selected and sent back to all workers for the next update. However, the updates of the local PG requires sharing of "active features" at each communication round. It is unknown how to communicate active features in a private and communication efficient way.

2) Uniform Serial Sampling: Lacoste-julien *et al.* [34] is a randomized BCFW method selecting the block to be updated in each iteration according to the uniform serial sampling, i.e., samples one block at each iteration with uniform probability. For analyzing the convergence, Lacoste-julien *et al.* [34] designed the step-size $\gamma^t = (2d/(t+2d))$ and introduced the product curvature to obtain $h^t \leq ((2d(C_f^\otimes + h^0))/(t+2d))$ primal gap. The product curvature $C_f^\otimes := \sum_{i=1}^d C_f^i$, where C_f^i is the blockwise partial curvature for measuring the nonlinearity on individual $\mathcal{M}_i$

$$C_f^i := \sup_{\substack{\boldsymbol{x},\boldsymbol{s}\in\mathcal{M},\\ \gamma\in[0,1]}} \frac{2}{\gamma^2}\Big(f(\boldsymbol{x}+\gamma(\boldsymbol{s}_{[i]}-\boldsymbol{x}_{[i]})) - f(\boldsymbol{x}) - \gamma\langle \boldsymbol{s}_{(i)}-\boldsymbol{x}_{(i)}, \nabla_{(i)} f(\boldsymbol{x})\rangle\Big). \tag{2}$$

3) τ-Nice Sampling: AP-BCFW [33] is a parallel and distributed BCFW method, provided that all user nodes have the full replication of the entire dataset. During one communication pass, each worker uniformly samples one block from *all* blocks for updating and the server node summarizes τ nonduplicate updates (discard duplicate update, e.g., two workers sample the same node). Under ideal computational facility, Wang [33] analyzed the convergence by equalizing one communication pass as one iteration of centralized BCFW selecting blocks for updating according to τ-nice sampling, i.e., samples τ blocks with uniform probability. It requires yet another set of step-size $\gamma^t = (2d\tau/(\tau^2 t + 2d))$ and expected set curvature $C_f^\tau := \binom{d}{\tau}^{-1} \sum_{S\subset[d],|S|=\tau} C_f^{(S)}$, where the set curvature $C_f^{(S)}$ is

$$C_f^S := \sup_{\substack{\boldsymbol{x},\boldsymbol{s}\in\mathcal{M},\\ \gamma\in[0,1],|S|=\tau}} \frac{2}{\gamma^2}\Big(f(\boldsymbol{x}+\gamma(\boldsymbol{s}_{[S]}-\boldsymbol{x}_{[S]})) - f(\boldsymbol{x}) - \gamma\langle \boldsymbol{s}_{(S)}-\boldsymbol{x}_{(S)}, \nabla_{(S)} f(\boldsymbol{x})\rangle\Big). \tag{3}$$

AP-BCFW is obviously unsuited to the distributed feature set, since it would require copy-and-paste local features to other nodes before computation, which incurs high communication cost and raises privacy concern.

4) (K, τ)-Distributed Sampling: This is probably the simplest sampling scheme for the distributed optimization tasks with disjointedly divided local blocks, where each user nodes uniformly sampling τ blocks from their *local* blocks which collectively constitutes $K \times \tau$ random block updates from K workers. However, existing random BCFWs do not have convergence guarantee even for this simplest sampling.

5) Arbitrary Sampling: We consider a general BCFW that has guaranteed convergence under arbitrary sampling with two minimal assumptions: 1) the sampling is independent across iterations, i.e., the sampling distribution at the present iteration is independent of the sampling of the last iteration and 2) the sampling is proper that any block has nonzero probability to be sampled.

B. Differential Privacy (DP)

The definition of DP [9] for a randomized algorithm $\mathcal{ALG}$ with parameter ϵ, δ is formalized by

Definition 1: (ϵ, δ)-Differential Privacy ((ϵ, δ)-DP) A randomized algorithm $\mathcal{ALG}$ is (ϵ, δ)-differentially private if, for all neighboring datasets $\mathcal{D}$ and $\mathcal{D}'$, which differ in only one data sample and for all outputs $\mathcal{O}$ we have $Pr(\mathcal{ALG}(\mathcal{D}) \in \mathcal{O}) \leq e^{\epsilon} Pr(\mathcal{ALG}(\mathcal{D}') \in \mathcal{O}) + \delta$.

DP has composition properties, which enable a DP algorithm to be constructed from building blocks and be applicable to algorithms progressing over multiple iterations. For details, refer to Lemmas A and B in the Supplementary Material.

C. Differentially Private Optimization for Featurewise Distributed Dataset

1) Featurewise Distributed Private Optimization: The featurewise distributed data is more challenging than the samplewise distributed data under privacy restriction. For the samplewise distributed setting, each user node has enough information to take local updates (e.g., users can compute the local gradient based on their local data samples), and only the decision variables need to be communicated. However, for the featurewise distributed data, apart from the decision variable, additional information is required to be shared to perform local updates (e.g., computing the local partial blockwise gradient). In general, as more information sent by the user node, it is more likely that sensitive individual privacy is at risk, which makes the privacy protection design more challenging. As a largely unexplored setting, to the best of our knowledge, the very recent [45] is the only existing work that has considered the same differentially private ERM learning task with disjoint features held by different parties. They propose to add privacy protection during preprocessing by communicating perturbed sketched features [47] based on the private Johnson–Lindenstrauss transform, i.e., Lemmas C and D in the Supplement Material. Although the uplink communication is one-shot during the preprocessing and its sketching step partially relieves the high communication complexity in terms of the feature dimension d, its complexity is still linearly dependent on the sample size n [i.e., $O(n)$].

2) Private Frank-Wolfe Algorithm: Talwar *et al.* [31] proposed a centralized private FW for the ERM problem constrained by an atomic norm. In each iteration, the FW algorithm greedily selects an LO from the atomic norm set $\mathcal{A}$ (has a finite number of atomic norms) by picking the one with the largest duality gap. Talwar *et al.* [31] selected the iterative LO by the "Report-Noisy-Max" mechanism [9] (a particular variant of the more general exponential mechanism), which ensures the DP. For the LASSO task, [31] is proved to provide nearly optimal utility guarantee. Since the utility guarantee is based on the convergence analysis, the adaptation of the method to a distributed setting is nontrivial due to the missing convergence result for BCFW-AS. Furthermore, with features distributed among user nodes, apart from the LO evaluation, the gradient computation also requires additional perturbation for privacy protection, whose effect on utility demands careful quantization and further analysis.

Definition 2 (Report-Noisy-Max): Given a dataset $\mathbb{D}$, denote a collection of P functions $\upsilon_1, \ldots, \upsilon_p, \ldots, \upsilon_P$ defined on $\mathbb{D}$ with ℓ_1-sensitivity ι. The Report-Noisy-Max selects the index whose perturbed value function is the maximum. That is: $\hat{p}^* = \arg\max_{p \in [P]} \hat{\upsilon}_p$, where $\hat{\upsilon}_p = \upsilon_p + \text{pert}$ with $\text{pert} \sim \text{Lap}(\iota/\epsilon)$.

Lemma 1 (Privacy Guarantee for Report-Noisy-Max Mechanism [9]): The Report-Noisy-Max mechanism preserves $(2\epsilon, 0)$-DP.

D. Communication-Efficient Distributed Optimization for ERM

Communication cost has been recognized as the bottleneck issue on the scalability of the distributed optimization algorithms. Significant research attention is drawn to reduce the communication cost for the samplewise distributed setting. Among the latest progress, the quantized SGD methods [19]–[21] have achieved state-of-the-art communication efficiency in terms of the communication compression rate. In each iteration, the user nodes compress the stochastic gradient to low-precision representation by quantization operators before sending it to the server. For example, signSGD [21]–[24] achieves the most aggressive communication reduction ratio of 96.8% by compressing each element of the gradient to $\{-1, +1\}$. In particular, QFW [46] utilizes the quantization to reduce the communication cost for the Frank–Wolfe algorithm, which also applied only to the samplewise distributed setting.

DP has been incorporated with the quantized SGD. cpSGD [15] proposes a two-step quantized SGD with DP protection, which first uses the stochastic k-level quantization to compress the gradient, then adds the DP protection by the binomial mechanism perturbation. vqSGD [42] designs a DP quantization to unify the compression and perturbation steps together. Very recently, DP-signSGD [17] uses a randomized sign operator to make signSGD algorithm differentially private.

Sketching is also a popular technique to reduce the communication cost [43], [44], which randomly reduces a long uplink message into a smaller length. These methods share some similarities with our work since our private Johnson–Lindenstrauss transform is also a general type of sketching. However, their method is based on SGD to sketch the gradient, while ours is a coordinate descent method where we sketch the active feature. In particular, DP-FedAvg [44] proposes a differentially private federated averaging algorithm for the FL setting, which applies the CountSketch to the gradient. Again, these methods only apply to the samplewise distributed setting.

E. Summary and Comparison of Related Work

Table II summarizes the related work into four categories based on types of techniques: signSGD-related, quantized SGD-related, sketching-related and FW-related. We have the following discussions.

Discussion 1: (Comparing Our Method With Related Work)

TABLE II
SUMMARY AND COMPARISON OF RELATED WORK

Algorithm Name	Representative Papers	Differentially private	Uplink Communication Efficient	Feature-wise Distributed
signSGD	[22]–[24]	✗	✓	✗
DP-signSGD	[17]	✓	✓	✗
Quantization SGD	[19], [20], [41]	✗	✓	✗
cpSGD	[15]	✓	✓	✗
vqSGD	[42]	✓	✓	✗
Sketched-SGD	[43]	✗	✓	✗
DP-FedAvg with Sketching	[44]	✓	✓	✗
PRIDE	[45]	✓	✗	✓
DP-FW	[31]	✓	Not Applicable	Centralized
QFW (quantized FW)	[46]	✗	✓	✗
AP-BCFW (τ-nice sampling)	[33]	✗	✓	✗
dFW (elementary sampling)	[32]	✗	✓	✓
DP-BCFW-AS	This paper	✓	✓	✓

1) Under the samplewise distributed setting, there emerge many algorithms following different lines of techniques to achieve both DP and communication-efficiency, while under the featurewise distributed setting, our method is the only one that has considered both (i.e., no DP method ever considers the communication efficiency).
2) State-of-the-art communication efficient methods are based on the gradient compressed SGD (e.g., quantized SGD and signSGD), which aggressively save uplink communication by over 96.8%. As will be seen, our method achieves a similar aggressive uplink communication compression ratio based on a different technique, which is able to save the communication and protects DP simultaneously.
3) Sketching is also a promising technique for reducing the communication cost. Most recently, Liu *et al.* [44] proposed a CountSketch-based method for achieving communication efficiency and DP at the same time, which is the closet result with ours in terms of the design philosophy. However, there are three key distinctions: 1) they consider the samplewise distributed setting; 2) they apply the sketch on the difference of the decision variables, while we applied on the active feature; and 3) their design crucially relies on the specific CountSketch, while ours is the more general JL-transform, which includes many different sketching techniques.
4) Compared with Frank–Wolfe methods, ours has the most general algorithm design and convergence guarantee. In addition, our method achieves the same communication efficiency in terms of the communication compression ratio by a different technique than the quantization-based method [46]. In fact, our design of the JL-transform for reducing the communication cost itself is new to the FW literature, even without the privacy mechanisms. Finally, our method provides rigorous DP protection without deteriorating the utility.

III. BCFW-AS

A. Problem Formulation

We consider the sparse optimization problem

$$\arg\min_{\|\boldsymbol{x}\|_1\leq\eta} \sum_{i=1}^{n} f(\boldsymbol{x};\mathbb{D}_i) \tag{4}$$

where $\mathbb{D}_i$ represents the ith ($i = 1,\ldots,n$) sample of the $n \times d$ dataset $\mathbb{D}$, n is the sample size, and d is the feature dimension. With the featurewise distributed setting, the $n \times d$ data matrix is partitioned vertically with each disjoint partition held by one of K user nodes. $f(\boldsymbol{x})$ is a smooth convex loss function, e.g., least square loss, logistic loss, and smoothed hinge loss. The ℓ_1 norm is coordinate separable, meaning each coordinate component can be computed independently: $\|\boldsymbol{x}\|_1 = \sum_{j=1}^{d} |\boldsymbol{x}_j|$, where $|\cdot|$ computes the absolute value of the jth component of $\boldsymbol{x}$. For notational simplicity, we denote the constraint set as $\mathcal{M} = \mathcal{M}_1\times,\ldots,\times\mathcal{M}_d$. This notation is also for generality: our convergence result in Section III is not limited to the ℓ_1-norm constraint but also applies to more general coordinate separable convex compact $\mathcal{M}$.

B. Algorithm Overview

Our BCFW-AS enjoys greater generality in both the design and convergence guarantee, which include the following three aspects: 1) applicable to arbitrary proper sampling distribution; 2) introducing a new universal step-size choice; and 3) supporting approximate PG computation and approximate partial linear oracle evaluation. Correspondingly, we introduce the following new developments.

For 1), we introduce a new notion called the "expected curvature," which compactly associates the directional curvatures of the loss function with the sampling distribution of the BCFW algorithm. In essence, the curvature reflects the largest deviation of the loss function $f(\boldsymbol{x})$ from its linear approximation on the constraint set $\mathcal{M}$. Previous methods propose to calculate the curvature along some *particular sets of coordinates*, which captures the largest deviation deterministically without considering the sampling distribution. In contrast, the expected curvature takes the sampling distribution into consideration, which measures the largest deviation "averaged" over all choices of *random sets of coordinates* that are sampled during the BCFW execution. This way, it manifests the interaction between the geometric property along different directions and the probability of these directions being sampled.

For 2), we observe that the existing designs of the step-size is specific to the sampling distribution they use. The step-size designed for one sampling distribution cannot be easily adapted to another. We discover that the proper step-size crucially relies on the minimum entry of the sampling probability (denoted by $p_{\min}$). To this end, we design the step-size to

be $\gamma^t = (2/(p_{\min} \cdot t + 2))$, where γ^t is the step-size at iteration t. For arbitrary samplings, this universal step-size still guarantees the convergence of the algorithm without a sampling-by-sampling redesign, which will be shown in Theorems 1 and 2.

For 3), the PG and the partial LO are two major steps of BCFW and computed based on the raw data. We allow both computations to be evaluated approximately, where the approximation ratios are measured by the parameters $\varrho_g \geq 0$ for PG and $\varrho_l \geq 0$ for LO, respectively. It includes the BCFW with exact PG and LO computations as a special case, which corresponds to $\varrho_g = 0$ and $\varrho_l = 0$. In addition to the generality, the approximations have two more significant benefits. First, it often costs lower computational complexity than the exact evaluation in practice [48]. Second, since both steps are directly based on the raw data, the approximations will help accommodate the perturbations injected to both steps for privacy-preserving purpose, as will be seen in Section IV. Under mild assumptions, we will show that BCFW-AS is guaranteed to converge even with the approximations, as will be seen in Theorem 2.

C. Expected Curvature

The expected curvature is formally defined as follows.

Definition 3: (Expected Curvature) The expected curvature of $f(\boldsymbol{x})$ with arbitrary proper sampling $\mathcal{S}$ is defined as

$$C_f^{\mathbb{E}\mathcal{S}} = \sup_{\substack{\boldsymbol{x},\boldsymbol{s}\in\mathcal{M},\\ \gamma\in[0,1]}} \mathbb{E}_{\mathcal{T}\sim\mathcal{S}}\Big[\frac{2}{\gamma^2} f(\boldsymbol{x}+\gamma(\boldsymbol{s}_{[\mathcal{T}]}-\boldsymbol{x}_{[\mathcal{T}]})) - f(\boldsymbol{x}) - \gamma\langle \boldsymbol{s}_{[\mathcal{T}]}-\boldsymbol{x}_{[\mathcal{T}]}, \nabla f(\boldsymbol{x})\rangle\Big]. \quad (5)$$

Discussion 2: Let the probability of sampling the subset $\mathcal{T}$ be $p(\mathcal{T})$. Equation (5) associates the directional curvature along $\mathcal{T}$ with its probability $p(\mathcal{T})$ by $p(\mathcal{T}) \cdot [(2/\gamma^2) f(\boldsymbol{x} + \gamma(\boldsymbol{s}_{[\mathcal{T}]} - \boldsymbol{x}_{[\mathcal{T}]})) - f(\boldsymbol{x}) - \gamma\langle \boldsymbol{s}_{[\mathcal{T}]} - \boldsymbol{x}_{[\mathcal{T}]}, \nabla f(\boldsymbol{x})\rangle]$, and averages it over combinations of directions $\mathcal{T} \sim \mathcal{S}$ that are possible under $\mathcal{S}$. The final expected curvature achieves the supreme by accounting both the directional curvatures and their probabilities being sampled. Intuitively, under the arbitrary sampling $\mathcal{S}$, expected curvature averages deviation between the loss function and its linear approximation over all combinations of proper $\boldsymbol{x}, \boldsymbol{s}, \gamma$ and picks the "averaged largest pair," which ensures the overall supremacy along the majority of the directions. As for the rare case where the "averaged largest pair" does not capture the largest curvature precisely, the probability is too small for them to affect the overall convergence. In other words, BCFW-AS allows "bad updates" to happen occasionally, as long as their probabilities to be sampled are low.

In Propositions 1–4, we compare the expected curvature with: 1) Lipschitz smoothness ESO in [35], [36], and [38] and 2) existing hand-crafted curvatures for specific samplings. These are well-studied geometric quantities used by even the latest coordinate descent algorithms [49]. Through the comparison, we show that the expected curvature is not only reasonable in definition but also more suitable for the FW-type algorithm. The proofs are in Appendix E in the Supplementary Material for completeness. To ease the comparison, we follow the compared methods and assume the following.

Assumption 1: There is an $n \times d$ matrix $\boldsymbol{A}$ (e.g., data matrix) such that for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{M}$

$$f(\boldsymbol{y}) \leq f(\boldsymbol{x}) + \langle \boldsymbol{y}-\boldsymbol{x}, \nabla f(\boldsymbol{x})\rangle + \frac{1}{2}(\boldsymbol{y}-\boldsymbol{x})^\top \boldsymbol{A}^\top \boldsymbol{A}(\boldsymbol{y}-\boldsymbol{x}).$$

1) Comparison With ESO: The expected curvature is upper bounded by the ESO quantity times the squared diameter.

Proposition 1: Under Assumption 1, let us denote the pairwise probability matrix of arbitrary sampling $\mathcal{S}$ by $\mathbb{P}$, the diameter of $\mathcal{M}_i$ by $D_{\mathcal{M}_i}$. By choosing $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_d)$, where $\beta_i = \min\{\sigma'(\mathbb{P}), \sigma'(\boldsymbol{A}^\top\boldsymbol{A})\}\|A_i\|_2^2$, where $\sigma'(\mathbb{P}), \sigma'(\boldsymbol{A}^\top\boldsymbol{A})$ are the largest normalized eigenvalues of the matrices $\mathbb{P}$ and $\boldsymbol{A}^\top\boldsymbol{A}$, then $C_f^{\mathbb{E}\mathcal{S}} \leq \sum_{i=1}^d p_i \beta_i D_{M_i}^2$.

In the above proposition, β_i is exactly one of the ESO quantities obtained in [38] under the same assumption. Qu and Richtárik [38] have also provided many other estimation of β_i. We omit those comparisons because we can also show the same result in a similar fashion.

2) Comparison With Existing Curvatures: The expected curvature is a tighter estimation of the geometry constant compared to existing case-by-case designed curvatures. Due to this tightness, the convergence speed obtained by BCFW-AS will also be tighter, as will be shown in Remark 3 in Section III-E.

Proposition 2: Recall the global curvature C_f of [32] and [40] under elementary sampling, the product curvature $C_f^\otimes$ of [34] under uniform serial sampling, and expected set curvature C_f^τ of [33] under τ-nice sampling as introduced in Section II-A. Then, the following relationships hold:

$$C_f^{\mathbb{E}\text{element}} = C_f,\ C_f^{\mathbb{E}\text{uni·seri}} \leq \frac{1}{d} C_f^\otimes,\ C_f^{\mathbb{E}\tau\text{nice}} \leq C_f^\tau \quad (6)$$

where all the left-hand side terms denote the expected curvature under the corresponding samplings.

By Proposition 2, the expected curvature is always upper bounded by existing specific curvature constants introduced for specific samplings, which will result in refined convergence results as shown in Section III-C3.

3) Computing Expected Curvature Given Samplings: At first glance, the definition of the expected curvature seems abstract. Yet, as long as the sampling is given, we show that an accurate estimation of the expected curvature can be calculated. For illustration, we present the calculation of expected curvature under τ-nice sampling and (K, τ)-distributed sampling.

Proposition 3: Under Assumption 1, the expected curvature with τ-nice sampling, denoted by $C_f^{\mathbb{E}\tau nice}$, satisfies: $C_f^{\mathbb{E}\tau nice} \leq \tau\mu_1 + \tau(\tau-1)\mu_2$, where $\mu_1 = \sup_{i\in[d]} \|\boldsymbol{A}_i(\boldsymbol{s}_i - \boldsymbol{x}_i)\|_2^2$, $\mu_2 = \sup_{i,j\in[d], i\neq j} (\boldsymbol{A}_i(\boldsymbol{s}_i - \boldsymbol{x}_j))^\top (\boldsymbol{A}_j(\boldsymbol{s}_i - \boldsymbol{x}_j))$.

The above estimation matches the expected set curvature in [33], thus reconfirming the comparison in Proposition 2.

Proposition 4: Under Assumption 1, the expected curvature with (K, τ)-distributed sampling, denoted by $C_f^{\mathbb{E}(K,\tau)}$, satisfies $C_f^{\mathbb{E}(K,\tau)} \leq K\tau\mu_1 + K\tau(\tau-1)\mu_2 + K(K-1)\tau^2\mu_3$, where $\mu_1 = \sup_{i\in[d]} \|\boldsymbol{A}_i(\boldsymbol{s}_i - \boldsymbol{x}_i)\|_2^2$, $\mu_2 = \sup_{i,j\in\mathcal{P}_k, i\neq j} (\boldsymbol{A}_i(\boldsymbol{s}_i - \boldsymbol{x}_j))^\top (\boldsymbol{A}_j(\boldsymbol{s}_i - \boldsymbol{x}_j))$, $\mu_3 = \sup_{i\in\mathcal{P}_{k_1}, j\in\mathcal{P}_{k_2}, k_1\neq k_2} (\boldsymbol{A}_i(\boldsymbol{s}_i - \boldsymbol{x}_j))^\top (\boldsymbol{A}_j(\boldsymbol{s}_i - \boldsymbol{x}_j))$.

Finally, let us conclude the expected curvature subsection with the following discussion.

Remark 1: As pointed out in [40], the curvature constant is vital in its own stand.

1) It is more intrinsic to the geometric property of the loss function on the constraint set. For example, it can be a much tighter geometry constant than the common Lipschitz smoothness related quantity.
2) It is affine invariant, so it applies to arbitrary choices of norms. As a result, our approach via introducing the expected curvature is a fundamentally new development to the FW-type algorithms.

D. Algorithm Description

Algorithm 1 presents the BCFW-AS algorithm under the arbitrary proper sampling of $\mathcal{S}$. For notational convenience, in this section, $i \in [d]$ can be either a single coordinate or a block of coordinates that we do not explicitly differentiate the two meanings with an additional notation. In Section IV, i will refer to a single coordinate. For the sampling $\mathcal{S}$, we denote the probability of the sampling block i by p_i and collectively the probability vector of sampling each block by $\boldsymbol{p} := \{p_1, \ldots, p_d\}$. Let $p_{\min}$ denote the smallest entry in $\boldsymbol{p}$.

We use the superscript t to denote the number of iterations. In each iteration, line 3 samples a random set of blocks $\mathcal{T}^t$ from $\{1, 2, \ldots, d\}$ according to the sampling distribution $\mathcal{S}$; line 5 computes the approximate PG $\hat{\nabla}_{(i)} f(\boldsymbol{x})$ with inexactness parameter ϱ_g^i; line 6 evaluates the approximate partial LO $\hat{\boldsymbol{s}}_{(i)}$ with the inexactness parameter ϱ_l^i; line 7 updates the block ith decision variable for the sampled blocks $i \in \mathcal{T}^t$ with the step-size $\gamma^t = (2/(p_{\min} \cdot t + 2))$. Concisely, we can add up $\hat{\boldsymbol{s}}_i^t$ for all $i \in \mathcal{T}^t$ to denote $\hat{\boldsymbol{s}}_{[\mathcal{T}^t]}^t = \sum_{i \in \mathcal{T}^t} \hat{\boldsymbol{s}}_{[i]}^t$. Then, the overall update across all sampled blocks can be summarized as $\boldsymbol{x}_{[\mathcal{T}^t]}^{t+1} = \boldsymbol{x}^t + \gamma^t(\hat{\boldsymbol{s}}_{[\mathcal{T}^t]}^t - \boldsymbol{x}_{[\mathcal{T}^t]}^t)$.

Algorithm 1 BCFW Algorithm Under Arbitrary Sampling

1: **Input:** Initial feasible variable $\boldsymbol{x}^0$, step sequence γ^t, sampling distribution $\mathcal{S}$, inexactness parameters ϱ_g^i and ϱ_l^i, maximum iteration T;
2: **for** $t = 0, 1, \ldots, T-1$ **do**
3: Generate a random set $\mathcal{T}^t \subset [d]$ according to the sampling distribution $\mathcal{S}$;
4: **for each** $i \in \mathcal{T}^t$ **do**
5: Compute the approximate partial gradient $\hat{\nabla}_{(i)} f(\boldsymbol{x}^t)$, which satisfies eq.(8);
6: Compute the approximate partial linear oracle $\hat{\boldsymbol{s}}_{(i)}^t$, which satisfies eq.(9);
7: Update $\boldsymbol{x}_{(i)}^{t+1} = \boldsymbol{x}_{(i)}^t + \gamma^t(\hat{\boldsymbol{s}}_{(i)}^t - \boldsymbol{x}_{(i)}^t)$;
8: **end for**
9: **end for**
10: **Output:** $\boldsymbol{x}^T$;

E. Convergence Analysis for BCFW With Arbitrary Sampling

In this section, we derive the convergence results for both the exact and the inexact BCFW-AS algorithm. For the former, it provides a direct comparison to the existing case-by-case crafted BCFW algorithms under specific samplings, which verifies its superiority in terms of the generality, tightness in convergence, and easiness to use. For the latter, the inexactness provides the opportunity for further gradient compression and privacy perturbation, which are essential to the development of communication efficient and differentially private distributed BCFW algorithms, as exemplified in Section IV.

TABLE III
COMPARISON OF BCFW CONVERGENCE RESULTS

Sampling	p_{min}	Existing result	Ours
Elementary	1	$\frac{2(1+\delta)C_f}{t+2}$ [32], [40]	$\frac{2(h_0+(1+\delta)C_f)}{t+2}$
Uniform Serial	$\frac{1}{d}$	$\frac{2d(h^0+(1+\delta)C_f^{\otimes})}{t+2d}$ [34]	$\mathbf{\frac{2d(h^0+(1+\delta)C_f^{\mathbb{E}uni\cdot seri})}{t+2d}}$
τ-nice	$\frac{\tau}{d}$	$\frac{2d(h^0+(1+\delta)C_f^{\tau})}{\tau^2 t+2d}$ [33]	$\mathbf{\frac{2d(\tau h^0+(1+\delta)C_f^{\mathbb{E}\tau nice})}{\tau^2 t+2d\tau}}$
(K,τ)-distributed	$\frac{(K\tau)^2}{d^2}$	unknown	$\mathbf{\frac{2d(K\tau h^0+(1+\delta)C_f^{\mathbb{E}(K,\tau)})}{(\tau K)^2 t+2dK\tau}}$
Arbitrary	p_{min}	unknown	$\mathbf{\frac{2(h^0+\frac{(1+\varrho)}{p_{min}}C_f^{\mathbb{E}\mathcal{S}})}{p_{min}\cdot t+2}}$

1) Convergence of BCFW-AS With Exact Update: Equipped with the expected curvature, we derive the convergence result of BCFW-AS with exact gradient and LO computation, which corresponds to $(\varrho_g^i, \varrho_l^i) = (0, 0), \forall i \in [1, d]$. It highlights the essential roles of the expected curvature and the universal step-size played in the algorithm design and the proof process.

Theorem 1 (Convergence Result of BCFW-AS With Exact Update): Let $D_{\mathcal{M}}$ denote the diameter of the constraint set $\mathcal{M}$, $|\mathcal{T}|$ denote the maximum sampling set size among iteration $1, \ldots, t$. Assume the calculation of the PG is accurate and the local LO is taken to be optimal, i.e., $(\varrho_g^i, \varrho_l^i) = (0, 0), \forall i \in [1, d]$. For each iteration $t \geq 0$, the iterates $\boldsymbol{x}^t$ generated by Algorithm 1 with step-size $\gamma^t = \frac{2}{p_{\min} \cdot t + 2}$ satisfy

$$\mathbb{E}[f(\boldsymbol{x}^t)] - f(\boldsymbol{x}^*) \leq \frac{2(h^0 + (C_f^{\mathbb{E}\mathcal{S}}/p_{\min}))}{p_{\min} \cdot t + 2} \tag{7}$$

where $\boldsymbol{x}^*$ denotes an optimum of the problem, $h^0 := f(\boldsymbol{x}^0) - f(\boldsymbol{x}^*)$, $p_{\min}$ denotes the minimum entry of probability vector $\boldsymbol{p}$, $C_f^{\mathbb{E}\mathcal{S}}$ is the expected curvature for arbitrary sampling $\mathcal{S}$.

Discussion 3: Theorem 1 shows the superiority of the BCFW-AS compared to existing BCFW algorithms with particular samplings: 1) BCFW-AS recovers all existing BCFW algorithms by simply substituting $p_{\min}$ into the universal step-size; 2) BCFW-AS provides convergence results for all existing methods with comparable or even tighter rates; and 3) BCFW-AS enables the adoption of more interesting new sampling schemes like (K, τ)-distributed sampling and even importance sampling, stratified sample, and so on.

We summarize the comparison in Table III with better results highlighted in **bold**. For each row, it provides the comparison between the convergence results achieved by our BCFW-AS and the existing methods under the same algorithm-independent conditions, including sampling distribution, initial primal gap of h^0, data dimension of d, n, and approximation parameter of δ. Table III indicates that: 1) ours

has the best generality to be applied to all types of sampling distributions, including the new (K, τ)-distributed sampling and the arbitrary sampling; 2) under the elementary sampling, we achieve the same convergence rate of $O(1/t)$ and the same dependence on the curvature C_f; 3) under the uniform serial sampling, we obtained a tighter convergence, since our expected curvature constant can be smaller, as indicated by Proposition 2; and 4) under the τ-nice sampling, we recover an overall compatible convergence rate, where our dependence on the curvature-related term is better, while the dependence on the initial primal gap is inferior.

2) Convergence of BCFW-AS With (ϱ_g, ϱ_l)-Approximation: Next, we extend the convergence guarantee to the inexact BCFW-AS which evaluates ϱ_g-approximate PG and ϱ_l-approximate LO, which are formalized by Assumption 2.

Assumption 2 (Inexact LO and Inexact Gradient): Let γ^t denote the step-size at iteration t, $C_f^{\mathbb{E}\mathcal{S}}$ denote expected curvature of function f with sampling $\mathcal{S}$.

1) Let ϱ_g^i be the inexact gradient constant parameter and $\nabla_{(i)} f(\boldsymbol{x})$ be the exact PG at $\boldsymbol{x}$. The PG $\hat{\nabla}_{(i)} f(\boldsymbol{x})$ satisfies

$$\|\hat{\nabla}_{(i)} f(\boldsymbol{x}) - \nabla_{(i)} f(\boldsymbol{x})\|^*_{(\mathcal{M}_i)} \leq \frac{\varrho_g^i \gamma^t C_f^{\mathbb{E}\mathcal{S}}}{2} \tag{8}$$

where $\|\cdot\|^*_{(\mathcal{M}_i)}$ is the dual norm defined on $\mathcal{M}_i$.

2) Let ϱ_l^i be a constant parameter. The inexact LO $\hat{\boldsymbol{s}}_{(i)}$ satisfies

$$\langle \hat{\boldsymbol{s}}_{(i)}, \hat{\nabla}_{(i)} f(\boldsymbol{x}) \rangle \leq \min_{\boldsymbol{s}_{(i)} \in \mathcal{M}_i} \langle \boldsymbol{s}_{(i)}, \hat{\nabla}_{(i)} f(\boldsymbol{x}) \rangle + \frac{\varrho_l^i \gamma^t C_f^{\mathbb{E}\mathcal{S}}}{2}. \tag{9}$$

Remark 2: Assumption 2 is a mild one. First, (8) means the inexact PG can be $(\varrho_g^i \gamma^t C_f^{\mathbb{E}\mathcal{S}}/2)$ away from the exact PG under the dual norm given by the constraint $\mathcal{M}_i$. For the ℓ_1 norm, it means the maximum norm between the inexact and the exact PG should not be too large. Second, (9) means the inexact LO $\hat{\boldsymbol{s}}_{(i)}$ does not need to exactly minimize $\min_{\boldsymbol{s}_{(i)} \in \mathcal{M}_i} \langle \boldsymbol{s}_{(i)}, \hat{\nabla}_{(i)} f(\boldsymbol{x}) \rangle$. It can be $(\varrho_l^i \gamma^t C_f^{\mathbb{E}\mathcal{S}}/2)$ larger than the minimum value. In Section IV, we will see that Assumption 2 suits well with the random perturbation of the privacy mechanisms and the approximation error of the Johnson–Lindenstrauss transform used for the communication compression. That is, the approximation errors introduced by privacy mechanisms automatically satisfy Assumption 2. Thus, we do not need to worry about the satisfactory of these assumptions at all, when apply the convergence result to our private algorithm.

Theorem 2 (Convergence Result of BCFW-AS With (ϱ_g, ϱ_l)-Approximations): Under Assumption 2, let $D_{\mathcal{M}}$ denote the diameter of constraint set $\mathcal{M}$ and $|\mathcal{T}|$ denote the maximum sampling set size among iteration $1, \ldots, t$. Take $\varrho_g = \max_i \varrho_g^i$, $\varrho_l = \max_i \varrho_l^i$, $\varrho = (D_{\mathcal{M}} + |\mathcal{T}|)\varrho_g + \varrho_l |\mathcal{T}|)$. For each $t \geq 0$, the iterates $\boldsymbol{x}^t$ generated by Algorithm 1 with step-size $\gamma^t = (2/(p_{\min} \cdot t + 2))$ satisfy

$$\mathbb{E}[f(\boldsymbol{x}^t)] - f(\boldsymbol{x}^*) \leq \frac{2(h^0 + ((1+\varrho)/p_{\min}) C_f^{\mathbb{E}\mathcal{S}})}{p_{\min} \cdot t + 2} \tag{10}$$

Algorithm 2 Differentially Private Inexact BCFW-AS With Distributed Features for Sparse Optimization

Input: Initial feasible variable $\boldsymbol{x}^0 = \boldsymbol{0}_d$, active feature aggregation $\hat{\boldsymbol{q}}^{-1} = \boldsymbol{0}_d$, step sequence γ^t, maximum iteration T, ℓ_1-norm ball size η, differential privacy parameters (ϵ, δ);

Server node initialize: broadcast JL-transform matrix $\boldsymbol{J}$;
User node initialize: receive JL-transform matrix $\boldsymbol{J}$;
for $t = 0, 1, \ldots, T-1$ **do**
  On Each User Node $k \in 1, \ldots, K$**:**
  Receive $\boldsymbol{x}^t$ and $\hat{\boldsymbol{q}}^{t-1}$ from Server;
  For each $i \in \mathcal{P}_k$, compute $\hat{\nabla}_i f(\boldsymbol{x}^t)$ by Table V;
  $v_i = \hat{\nabla}_i f(\boldsymbol{x}^t) + pert$;
  $i\hat{d}x_k^t = sign(v_i) \cdot \arg\max_{i \in \mathcal{P}_k} v_i$;
  Send $i\hat{d}x_k^t$ and $\hat{\boldsymbol{a}}_k^t = \boldsymbol{J}\boldsymbol{a}_{|i\hat{d}x_k^t|} + \boldsymbol{\xi}$ to Server;
  On Server Node:
  Wait and Receive $i\hat{d}x_k^t$ and $\hat{\boldsymbol{a}}_k^t$ from all K workers;
  $\hat{\boldsymbol{s}}^t = \eta \sum_{k=1}^K -sign(i\hat{d}x_k^t)\boldsymbol{e}_{[i\hat{d}x_k^t]}$;
  $\boldsymbol{x}^{t+1} = (1-\gamma^t)\boldsymbol{x}^t + \gamma^t \hat{\boldsymbol{s}}^t$;
  $\hat{\boldsymbol{q}}^t = (1-\gamma^{t-1})\boldsymbol{q}^{t-1} + \gamma^{t-1} \sum_{k=1}^K -sign(idx_k^t)\frac{\eta}{n}\hat{\boldsymbol{a}}^{t-1}_{|idx_k^t|}$;
  Broadcast $\boldsymbol{x}^{t+1}$ and $\hat{\boldsymbol{q}}^t$ to all K workers;
end for
Output: $\boldsymbol{x}^T$;

where $\boldsymbol{x}^*$ denotes an optimum of the problem, $h^0 := f(\boldsymbol{x}^0) - f(\boldsymbol{x}^*)$, $p_{\min}$ denotes the minimum entry of probability vector $\boldsymbol{p}$, $C_f^{\mathbb{E}\mathcal{S}}$ is the expected curvature for arbitrary sampling $\mathcal{S}$.

IV. Uplink Communication Efficient Differentially Private BCFW With Distributed Features

To exemplify the powerfulness of the inexact BCFW-AS, we consider the sparse optimization for featurewise distributed datasets with the uplink communication efficiency and the DP restrictions. Theorem 2 plays a crucial role in ensuring the convergence as well as analyzing the utility and uplink communication cost.

A. Featurewise Distributed Sparse Regression

For notational simplicity, we consider the following balanced featurewise distributed setting:

$$\boldsymbol{A} = [\overbrace{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{d/K}}^{\text{User 1}}, \ldots\ldots, \overbrace{\boldsymbol{a}_{d/K \cdot (K-1)+1}, \ldots, \boldsymbol{a}_d}^{\text{User } K}] \tag{11}$$

where the n samples by d features dataset $\boldsymbol{A}$ is evenly split among K user nodes with each holding d/K features $\boldsymbol{a}_i \in \mathbb{R}^{n \times 1}$. We assume the features are sparse, i.e., $nnz(\boldsymbol{a})$ is a constant and is much smaller than d, n, i.e., $nnz(\boldsymbol{a}) = O(d)$ and $nnz(\boldsymbol{a}) = O(n)$. We denote the index set associated with user k's features by $\mathcal{P}_k$.

B. Algorithm Overview

1) Key Steps in Basic Form: The server node will perform simple functionalities of aggregating user nodes' updates and broadcasting the aggregations back to user nodes, which is

light in computation and thus beneficial to the scalability. The major new designs will lie in the user nodes' side. We use the subscript $k \in \{1, \ldots, K\}$ to denote the local variables of node k and again use the superscript $t \in \{1, \ldots, T\}$ to denote the iteration number. According to the BCFW-AS update, each user node takes two key steps per-iteration.

1) *Partial LO Computation:* The partial LO is the signed index idx_k^t, which can be computed as

$$idx_k^t = -\text{sign}(\nabla_{i_k^t} f(\boldsymbol{x}^t)) \cdot i_k^t, \quad \text{where } i_k^t = \arg\max_{i \in \mathcal{P}_k} |\nabla_i f(\boldsymbol{x}^t)|. \tag{12}$$

2) *PG Computation:* Each user node evaluates $\nabla_{(\mathcal{P}_k)} f(\boldsymbol{x}^t)$ with respect to their own features. In the featurewise distributed setting, it cannot be evaluated independently solely based on local features without exchanging a global information. For example, for the least-square loss case, the computation by user node k is

$$\text{for each } i \in \mathcal{P}_k, \nabla_i f(\boldsymbol{x}^t) = \boldsymbol{a}_i^\top \left(\frac{1}{n}\boldsymbol{A}\boldsymbol{x}^t\right) - \boldsymbol{a}_i^\top \left(\frac{1}{n}\boldsymbol{y}\right) \tag{13}$$

where $\boldsymbol{A}_{\mathcal{P}_{1,\ldots,k-1,k+1,\ldots,K}}$ are kept on the other $K-1$ user nodes. This is the essential difficulty of the featurewise distributed setting compared to the samplewise distributed setting where the gradients can be evaluated solely based on the local samples.

2) Uplink Communication-Efficient Designs: The communication of idk_k^t is extremely communication efficient, which costs only 1-bit for the sign and one unsigned integer for the index. We focus on the uplink communication-efficient design to reduce the communication cost for computing the PG. We propose a two-level uplink communication reduction mechanism.

1) *Level at the Number of Features:* We propose the "share-at-need" feature sharing strategy to communicate only the nonduplicate "active features" (i.e., the features need to be communicated, formally defined in Definition 4 later). At first glance, it seems that the distributed dataset $\boldsymbol{A}$ (of size $n \times d$) needs to be fully exchanged in order to compute the PG $\nabla_i f(\boldsymbol{x}^t)$ in (13), which causes high communication cost and severe privacy leakage. At the first level of communication reduction, we decrease the number of features to be communicated from d to at most T for each user node, where $T \ll d$ is the total number of iterations. Our strategy is to introduce an intermediate variable $\boldsymbol{q}^t = (1/n)\boldsymbol{A}\boldsymbol{x}^t$ and update it iteratively, which suffices to compute the PG in (13). The update of $\boldsymbol{q}^t$ requires only a subset of features (called active features), thus the user nodes only need to share the features at need.
2) *Level at the Length of One Feature:* We further reduce the length of the shared active features from n to m, where $m \ll n$. We introduce a private Johnson–Lindenstrauss transform, which sketches the active feature in a DP manner to simultaneously reduce the communication and protect the privacy.

3) DP Designs: Each user nodes need to send the signed index and the active feature through uplink communication iteratively, both of which are computed based on the users' data and thus require privacy protections. We design the following two privacy-preserving mechanisms.

1) *Private Signed Index Computation:* In essence, idx_k^t in (12) is a direction chosen from $2|\mathcal{P}_k|$ (i.e., $\pm i$, $i \in \mathcal{P}_k$) directions in a way that its associated value $\text{sign}(idx_k^t)\nabla_{|idx_k^t|} f(\boldsymbol{x}^t)$ is the largest among all $2|\mathcal{P}_k|$ values. In other words, the signed index idx_k^t gets reported if its associated value reaches the maximum. It suggests the correspondence with the "Report-noisy-max" mechanism in Definition 2. To protect the privacy, we will perturb the values (i.e., each element of the PG) by calibrated DP noise and select the direction $i\hat{d}x_k^t$ whose perturbed associate value reaches maximum.
2) *Private Active Feature Sharing:* In the algorithm description part, we will see that only the feature with the index $i\hat{d}x_k^t$ needs to be shared, which is the active feature at iteration t. As mentioned, we will utilize the private JL-transform by carefully calibrating the noise magnitude to balance the privacy and the utility.

C. Algorithm Description

The algorithm is summarized in Algorithm 2. The server side update is as simple as aggregating all the partial LOs to update the decision variable and broadcasting the updated decision variable and private active features back to all user nodes. The following are user nodes' side mechanisms.

1) Uplink Communication-Efficient Mechanisms: We introduce the intermediate variable $\boldsymbol{q}^t = (1/n)\boldsymbol{A}\boldsymbol{x}^t$, which can be updated iteratively

$$\boldsymbol{q}^t = (1 - \gamma^{t-1})\boldsymbol{q}^{t-1} + \gamma^{t-1} \sum_{k=1}^{K} -\text{sign}\left(i\hat{d}x_k^t\right)\frac{\eta}{n}\boldsymbol{a}_{|i\hat{d}x_k^t|}^{t-1}. \tag{14}$$

Definition 4 (Active Feature): The active feature sent by user k at iteration t is the feature whose index is $|i\hat{d}x_k^t|$. We denote it by $\boldsymbol{a}_{|i\hat{d}x_k^t|}$ and its differentially private counterpart by $\hat{\boldsymbol{a}}_{|i\hat{d}x_k^t|}$.

Equation (14) indicates each user node only need to send a single active feature per-iteration. Overall, no more than T nonduplicate active features are required for communicating across T iterations. The PG can then be computed by $\boldsymbol{q}^t$. For example, for the least square loss, it is

$$\text{for each } i \in \mathcal{P}_k, \nabla_i f(\boldsymbol{x}^t) = \boldsymbol{a}_i^\top(\boldsymbol{q}^t) - \boldsymbol{a}_i^\top \left(\frac{1}{n}\boldsymbol{y}\right) \tag{15}$$

while more general cases are deferred to Section IV-E.

2) DP Mechanisms: After identifying uplink communication variables of the index and active feature, we design privacy mechanisms to ensure they do not leak sensitive user information.

1) *Private Signed Index Computation:* To leverage the "Report-noisy-max" mechanism to preserve privacy for the computation of idx, instead of selecting the index based on clean associated value, we selects based on

the noise-perturbed associated value. It has the following procedures.

a) For each $i \in \mathcal{P}_k$, compute noise injected PG by $v_i = \text{sign}(i)\nabla_i f(\boldsymbol{x}) + \text{pert}$, where pert is a Laplacian noise

$$\text{pert} \sim \text{Lap}\left(\frac{(2G\eta/Kn)\cdot\sqrt{2T\log(1/(\delta/2))}}{\left(\frac{\epsilon}{2K}\right)/2}\right).$$

b) Pick $i\hat{d}x_k^t = \text{sign}(v_{i^*})i^*$, where $v_{i^*} = \max_{i\in\mathcal{P}_k} v_i$.

2) *Private Active Feature Sharing:* We apply the private Johnson-Lindenstrauss transformation to the active features $\boldsymbol{a}^t_{|i\hat{d}x|} \in \boldsymbol{A}_{\text{trans}}$, given by

$$\hat{\boldsymbol{a}}^t_{|i\hat{d}x|} = \boldsymbol{J}\boldsymbol{a}^t_{|i\hat{d}x|} + \boldsymbol{\xi} \tag{16}$$

where $\boldsymbol{J}$ is an $m \times n$ Gaussian sketch matrix [50] (a type of Johnson-Lindenstrauss transformation matrix) and $\boldsymbol{\xi}$ is an $m \times 1$ perturbation vector. The perturbation is calibrated to be

$$\xi_j \sim \mathcal{N}(0, \pi^2),\ \pi = \frac{\sigma(\boldsymbol{J})\sqrt{KT}\sqrt{2\left(\ln\left(\frac{1}{\delta}\right)+\epsilon/2\right)}}{n\epsilon/2} \tag{17}$$

where $\sigma(\boldsymbol{J})$ is the leading singular value of $\boldsymbol{J}$ and $j \in 1, \ldots, m$.

Discussion 4 (Our Novelties and Strengths): 1) We extend the "Report-noisy-max" in the centralized Frank-Wolfe algorithm [31] to the featurewise distributed setting. Compared to the centralized case where only the signed index is inexact, the gradient is also inaccurate due to the perturbed active features. Our design and analysis ensure that our extension of the "Report-noisy-max" still guarantees the same nearly optimal utility as in the centralized setting.

2) The sketched active feature sharing for reducing the communication cost itself is new to the distributed FW algorithm, even without the privacy protection designs. Also, our analysis shows that with our designed m, Algorithm 2 is guaranteed to converge with the same $O(1/T)$ rate, i.e., communication reduction for free.

3) The sketching is also key to ensuring nearly optimal utility. Note that the gradient inexact parameter is the much smaller $O(\log(n)/n)$ with sketch (see Table IV and Theorem 4), compared to that of $O(\sqrt{n}/n)$ without sketch. Hence, the sketched features require much less noise for preserving DP. Otherwise, the optimal utility of $O((n)^{(2/3)})$ is impossible to be preserved with inexactness parameter as large as $O(\sqrt{n}/n)$.

D. Analysis of Algorithm 2

In the following, we analyze the DP, utility, and uplink communication cost of Algorithm 2.

TABLE IV
SUMMARIZATION OF PARAMETERS IN THEOREM 4

Param.	Formulation
ϱ_g	$O\left(\frac{((K\theta nnz(\boldsymbol{a})+K^{3/2}\sqrt{nnz(\boldsymbol{a})m}\sigma(\boldsymbol{J})^2)/p_{min})\sqrt{T}\sqrt{(\log(1/\delta)+\epsilon)}}{n\epsilon C_f^{\mathbb{E}(K,\tau)}\gamma^T(2\eta+K\tau)}\right)$
ϱ_l	$O\left(\frac{2G\eta\sqrt{32T\log(1/(\delta/2)}\log(2G\eta K\tau T)}{n\epsilon C_f^{\mathbb{E}(K,\tau)}\gamma^T(K\tau)}\right)$
C_g	$((K\eta\theta nnz(\boldsymbol{a})+K^{3/2}\sqrt{nnz(\boldsymbol{a})}\sigma(\boldsymbol{J})^2))\sqrt{(\log(1/\delta)+\epsilon)}$
C_l	$2G\eta\sqrt{32\log(1/(\delta/2)}$
m	$\Omega(\frac{1}{\theta^2}\log\frac{T}{\delta/2})$

1) DP: The sensitive information of the users are leaked via the uplink communications of the indices and active features. The following two lemmas show that the proposed privacy mechanisms are sufficient to guarantee DP.

Lemma 2: Private active feature sharing preserves $(\epsilon/2, \delta/2)$-DP.

Lemma 3: Private index computing preserves $(\epsilon/2, \delta/2)$-DP.

Then, the overall privacy guarantee can be obtained.

Theorem 3: Algorithm 2 is (ϵ, δ)-differentially private.

Proof: Applying serial composition property in Lemma A to compose Lemmas 2 and 3, Algorithm 2 is (ϵ, δ)-DP. □

2) Utility: In this section, we compare the utility of Algorithm 2 with the centralized DP Frank–Wolfe method [31], which proves that the nearly optimal utility for private LASSO is $O(1/n^{2/3})$. The utility is defined to be the excess empirical risk [10], [13], [31], as formalized in the following definition.

Definition 5: (Excess Empirical Risk [31]) Let the loss function be $f(\boldsymbol{x}) = (1/n)\sum_{i=1}^n f(\boldsymbol{x}; \mathbb{D}_i)$ and the constraint set be $\boldsymbol{x} \in \mathcal{M}$. For a differentially private algorithm $\mathcal{A}$ with output $\boldsymbol{x}_O$, the excess empirical risk $\mathcal{R}(\mathcal{A})$ is defined as

$$\mathcal{R}(\mathcal{A}) := \mathbb{E}_{\mathcal{A}}\left[\frac{1}{n}\sum_{i=1}^n f(\boldsymbol{x}_O; \mathbb{D}_i)\right] - \min_{\boldsymbol{x}\in\mathcal{M}}\frac{1}{n}\sum_{i=1}^n f(\boldsymbol{x}; \mathbb{D}_i) \tag{18}$$

where the expectation is taken with respect to the randomness of the differentially private algorithm $\mathcal{A}$.

The utility analysis is based on the convergence result of BCFW-AS developed in Section III with the sampling being (K, τ)-distributed sampling. As perturbation and sketching are introduced, the gradient and LO are inexact. The key step is to show Algorithm 2 satisfies Assumption 2 and show that $\hat{\boldsymbol{s}}^t$ and $\hat{\nabla} f(\boldsymbol{x}^t)$ are ϱ_l-LO and ϱ_g-PG correspondingly. The following theorem presents the utility guarantee and the parameters appeared are summarized in Table IV, where θ is the JL-transform parameter and $G = (1/n)\|\boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y})\|_\infty = O(1)$, $\tau = d/K$ is the number of features held by each user node.

Theorem 4: Let

$$T = \left(\frac{C_f^{\mathbb{E}(K,\tau)}n\epsilon}{C_g + C_l}\right)^{\frac{2}{3}}.$$

TABLE V
EXAMPLES OF PG UPDATES BY ACCUMULATED ACTIVE FEATURES

Loss type	Loss formulation	Partial gradient $\nabla_i f(\boldsymbol{x}^t)$	Partial gradient by $\hat{\boldsymbol{q}}^t$: $\hat{\nabla}_i f(\boldsymbol{x}^t)$
Least Square Loss	$\frac{1}{2n}\sum_{j=1}^{n}(y_j - \mathbb{D}_j\boldsymbol{x})^2$	$\boldsymbol{a}_i^\top(\frac{1}{n}\boldsymbol{A}\boldsymbol{x}^t) - \boldsymbol{a}_i^\top(\frac{1}{n}\boldsymbol{y})$	$(\boldsymbol{J}\boldsymbol{a}_i)^\top\hat{\boldsymbol{q}}^t - \boldsymbol{a}_i^\top\boldsymbol{y}$
Logistic Loss	$\frac{1}{n}\sum_{j=1}^{n}\log(1+\exp(-y_j\mathbb{D}_j\boldsymbol{x}))$	$-\boldsymbol{a}_i^\top\begin{bmatrix}\frac{y_1\exp(-y_1\mathbb{D}_1\boldsymbol{x}^t)}{1+\exp(-y_1\mathbb{D}_1\boldsymbol{x}^t)}\\ \cdots \\ \frac{y_n\exp(-y_n\mathbb{D}_n\boldsymbol{x}^t)}{1+\exp(-y_n\mathbb{D}_n\boldsymbol{x}^t)}\end{bmatrix}$	$-\boldsymbol{a}_i^\top\begin{bmatrix}\frac{y_1\exp(-y_1\boldsymbol{J}^\top\hat{\boldsymbol{q}}_1^t)}{1+\exp(-y_1\boldsymbol{J}^\top\hat{\boldsymbol{q}}_1^t)}\\ \cdots \\ \frac{y_n\exp(-y_n\boldsymbol{J}^\top\hat{\boldsymbol{q}}_n^t)}{1+\exp(-y_n\boldsymbol{J}^\top\hat{\boldsymbol{q}}_n^t)}\end{bmatrix}$
Smoothed Hinge Loss	$\frac{1}{2n}\sum_{j=1}^{n}\max\{0,1-y_j\mathbb{D}_j\boldsymbol{x}\}^2$	$-\boldsymbol{a}_i^\top\begin{bmatrix}y_1\max\{0,1-y_1\mathbb{D}_1\boldsymbol{x}^t\}\\ \cdots \\ y_n\max\{0,1-y_n\mathbb{D}_n\boldsymbol{x}^t\}\end{bmatrix}$	$-\boldsymbol{a}_i^\top\begin{bmatrix}y_1\max\{0,1-y_1\boldsymbol{J}^\top\hat{\boldsymbol{q}}_1^t\}\\ \cdots \\ y_n\max\{0,1-y_n\boldsymbol{J}^\top\hat{\boldsymbol{q}}_n^t\}\end{bmatrix}$

Algorithm 2 with least square loss update for the PG has the following privacy loss under the (ϵ, δ)-DP restriction:

$$\mathbb{E}[f(\boldsymbol{x}^T)] - \min_{x\in\mathcal{M}} f(\boldsymbol{x}) = O\left(\frac{(C_g + C_l)^{\frac{2}{3}}(C_f^{\mathbb{E}(K,\tau)})^{\frac{1}{3}}\log(2G\eta K\tau n)}{(n\epsilon)^{\frac{2}{3}}}\right).$$

Remark 3: According to Theorem 4, the utility is of order $O(1/n^{2/3})$, where $O(\cdot)$ hides constants and log factors, which is of the same order of the utility result $O(1/n^{2/3})$ of the centralized private FW method [31]. The utility result describes the relation of the privacy loss and the number of samples required. According to [31], this rate is nearly optimal meaning no algorithm can achieve smaller privacy risk given the same number of training samples expect a logarithm factor of n.

3) Uplink Communication Complexity: The uplink communication comes from: 1) KT *integers* for sending the private index (rather than the entire $d\times T$ *float* local decision variables x) and 2) $m\times KT$ for sending the private active features. With T in Theorem 4 and m in Table IV, we have Corollary 1.

Corollary 1: Algorithm 2 has uplink communication complexity $O((1/\theta)^2\log(n)^{1/3}K(n)^{2/3})$.

Remark 4: Compared to the one-shot communication at the preprocessing phase proposed by method [45], our uplink communication cost has better dependence on the sample size with $O(n^{2/3}\log(n)^{1/3})$ than theirs with $O(nr\log r)$, which shows our "share-at-need" feature sharing is more efficient than random sketching at preprocessing.

E. Extension to General Sparse ERM Optimization

In this section, we specify the computation of the approximate PG based on aggregated private active features $\hat{\boldsymbol{q}}^t$, with different choices of loss functions, which applies to least square loss, logistic loss, and squared hinge loss as examples.

The least-squares loss, the update is given in Section IV-A. For logistic loss, the term requiring global information sharing is

$$\left[\frac{\exp(-y_1\mathbb{D}_1\boldsymbol{x}^t)}{1+\exp(-y_1\mathbb{D}_1\boldsymbol{x}^t)},\ldots,\frac{\exp(-y_n\mathbb{D}_n\boldsymbol{x}^t)}{1+\exp(-y_n\mathbb{D}_n\boldsymbol{x}^t)}\right].$$

Again, $\boldsymbol{q}^t = \boldsymbol{A}\boldsymbol{x}^t$ is the sole term that requires to be communicated and can be done iteratively by active feature sharing. As for the private algorithm, the server side is the same as the private LASSO case in Algorithm 2. For the user node, after getting $\hat{\boldsymbol{q}}^t$, to computer the $\hat{\nabla}_i f(\boldsymbol{x}^t)$, it updates by

$$\hat{\nabla}_i f(\boldsymbol{x}^t) = \boldsymbol{a}_i^\top\left[\frac{y_1\exp(y_1\boldsymbol{J}^\top\hat{\boldsymbol{q}}_1^t)}{1+\exp(y_1\boldsymbol{J}^\top\hat{\boldsymbol{q}}_1^t)};\ldots;\frac{y_n\exp(y_n\boldsymbol{J}^\top\hat{\boldsymbol{q}}_n^t)}{1+\exp(y_n\boldsymbol{J}^\top\hat{\boldsymbol{q}}_n^t)}\right].$$

Similarly, for hinge loss

$$\hat{\nabla}_i f(\boldsymbol{x}^t) = \boldsymbol{a}_i^\top\left[y_1\max\{0,1+\boldsymbol{J}^\top\hat{\boldsymbol{q}}_1^t\};\ldots;y_n\max\{0,1+\boldsymbol{J}^\top\hat{\boldsymbol{q}}_n^t\}\right].$$

Finally, we summarize computation in Table V. As can be seen, to apply Algorithm 2 for different losses, it suffices to switch to corresponding computation of PG computed according to $\hat{\boldsymbol{q}}^t$.

V. EXPERIMENTAL RESULTS

In this section, we evaluate the following two aspects: 1) the convergence of the non-private BCFW-AS with a focus of different sampling distributions and 2) the performance of the differentially private BCFW-AS applied to the sparse optimization problem. To achieve this goal, we design six groups of experiments. In particular, groups 1 and 2 evaluate the nonprivate BCFW-AS, while groups 3–6 evaluate the differentially private BCFW-AS. In each group, a series of different choices of dataset generation parameters and algorithm settings are made, which are based on different purposes of the experiments. To this end, we will verify all the theoretical results one by one, and also study the parameter sensitivity of the proposed algorithm.

We use both synthetic datasets and real datasets. For all datasets, in order to best suit to the featurewise distributed setting, the number of features is larger than the number of samples. For the synthetic datasets, we generate the feature matrix $\boldsymbol{A}$ of size $[n, d]$ with a series of different feature sparsity ratios and the nonzero elements are from $\mathcal{N}(0,1)$. We generate the ground truth variable $\boldsymbol{x}$ with a variable sparsity ratios and the nonzero elements are from $\mathcal{N}(0,1)$. We let $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + noise$, where the elements of $noise$ are from $\mathcal{N}(0, 0.001)$. For the real datasets, we choose three datasets from Libsvm database [51]: colon-cancer, leukemia, and rcv1.

For each setting of the experiments, we randomly repeat its realization for 20 times and report the average results.

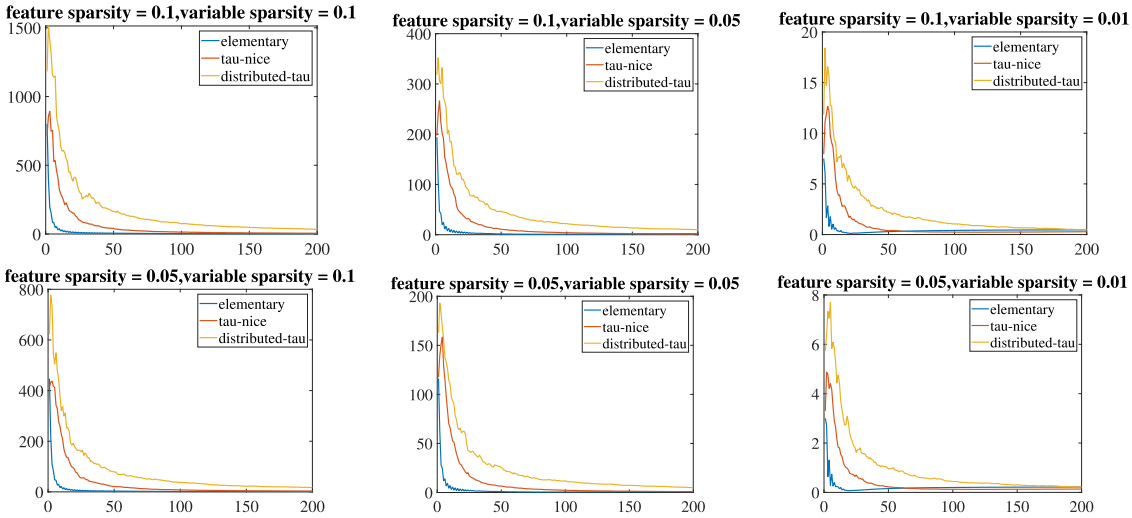

Fig. 1. Convergence of BCFW under different sampling distributions: objective value versus iteration number. The feature sparsity of the dataset varies from [0.1,0.01,0.001] (left to right) and the variable sparsity varies from [0.1,0.01,0.001] (top to bottom).

A. Comparison of BCFW-AS Under (K, τ)-Distributed Sampling With Known Simpler Samplings

First, we verify Theorems 1 and 2, especially the impact of $p_{\min}$ to the convergence, by comparing BCFW-AS with existing BCFW algorithms, each with different samplings: elementary, τ-nice, and (K, τ)-distributed. The former two have known convergence results, while the convergence of the last one comes from our Theorems 1 and 2. For τ-nice, we set τ to 4000, so that each feature has 0.4 probability of being sampled. For (K, τ)-distributed, we set $K = 2$ and keep τ as 4000. According to Theorem 1, since $p_{\min}$ is 1, 0.4, and 0.16 for elementary, τ-nice and (K, τ)-distributed accordingly, we expect elementary one to be the fastest while the (K, τ) one to be the slowest. Under six combinations of feature sparsity ratios and variable sparsity ratios, we run the FW algorithms with the different sampling distributions and plot the loss function value against the number of iterations. In Fig. 1, we plot the objective value versus the number of iterations. According to Fig. 1, BCFW is able to convergence under various samplings. Also, the convergence speed is related to the minimum sampling ratio, i.e., the larger the minimum sampling ratio, the faster the convergence speed. Therefore, the empirical evidence from Fig. 1 is consistent with Theorems 1 and 2.

B. Comparison of BCFW Under (K, τ)-Distributed Sampling With Varying K

In this section, we verify Theorems 1 and 2 with the different settings of K and sampling ratio. For the new convergence result of BCFW-AS with (K, τ)-distributed sampling, we study the convergence behavior of BCFW-AS with a varying number of K. We set the number of user nodes K to be 2, 4, 8, 16. For the same dataset, with increasing K, we expect the algorithm to converge faster, because more workers are working on it. In fact, BCFW algorithms can be viewed as greedy-type algorithms, which iteratively combine K atomic norms in every iteration until all relevant atomic norms are found. Thus, with K increasing, BCFW-AS converges faster. We plot the objective value versus the number of iterations in Fig. 2, which shows that the one with a larger K converges faster. This result is consistent with our expectation.

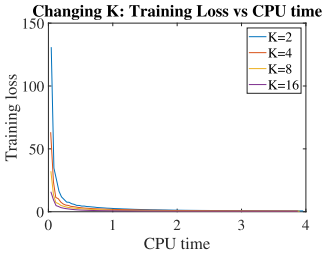

Fig. 2. Convergence of BCFW under (K, τ)-distributed sampling distributions with a series of K.

C. Comparison of DP-BCFW With Centralized DP-FW

In this section, we verify Theorem 4, especially the claim in Remark 3 that our distributed algorithm achieves the comparable utility as the centralized DP-FW [31]. We compare with centralized private algorithm [31] and use the nonprivate raw FW method as the baseline. The result is shown in Fig. 3. We use two sets of data dimensions $([d, n] = [20\,000, 2000], [2000, 200])$ and two sets of sparsity ratios (10% and 5%), resulting in four combinations of settings. For the distributed BCFW, we use $K = 4$ user nodes. We report the training loss with respect to the number of iterations. According to Fig. 3, both private algorithms converge slower than the nonprivate algorithm, and converge to a larger training loss. It is consistent with our expectation, because the private algorithms introduce perturbation. Furthermore, our distributed private FW is faster and converges at slightly better loss than the centralized one [31]. As a result, the private algorithm also enjoys the speedup with more workers as its nonprivate counterpart, which is shown in experiment group 2. With faster

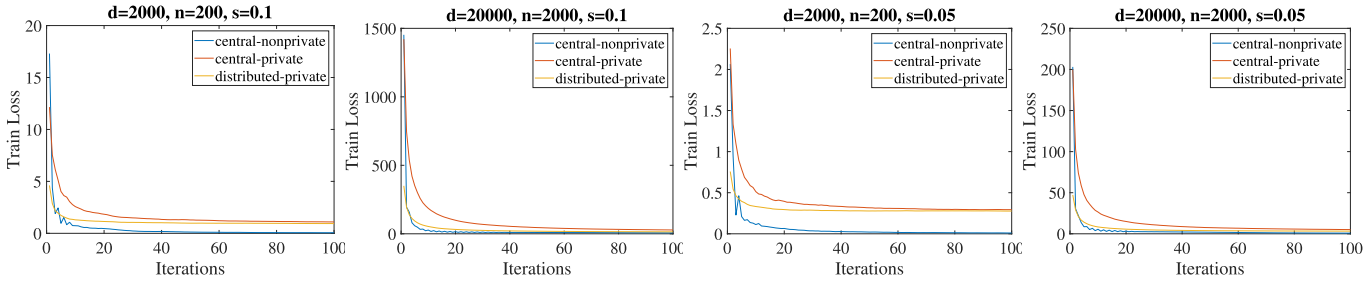

Fig. 3. Comparison of distributed DP-BCFW with centralized DP-FW with the centralized nonprivate FW as baseline.

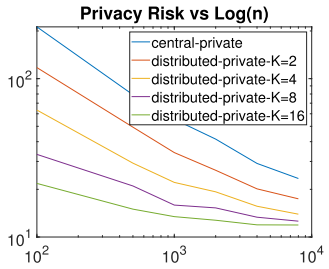

Fig. 4. Verifying Theorem 4 with privacy risk versus number of training samples n in log–log scale.

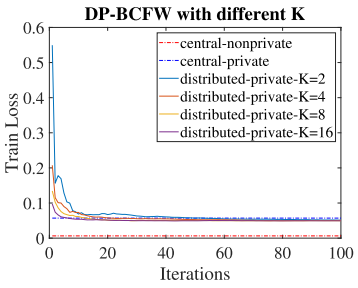

Fig. 5. Effect of changing the number of user nodes K.

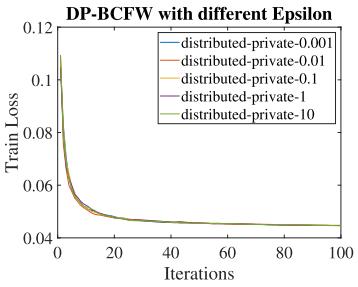

Fig. 6. Effect of changing the privacy budget ϵ.

convergence, less noise are accumulated to provide a slightly better accuracy.

D. Comparison of DP-BCFW With Different Number of Training Samples

In this part, we continue verifying Theorem 4, with a focus on the excess empirical risk dependence on the number of training data samples. We generate the dataset $\boldsymbol{A}$ containing 10 000 samples. The ground truth variable $\boldsymbol{x}$ and the response $\boldsymbol{y}$ follow the same distribution given in the beginning of Section V. The sparsity ratio is set to 1%.

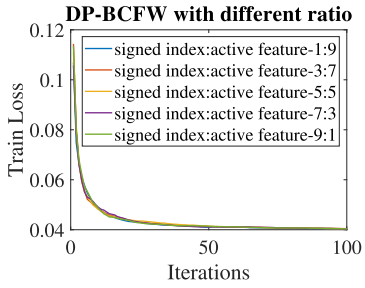

Fig. 7. Effect of changing the ratio of the privacy budget ϵ.

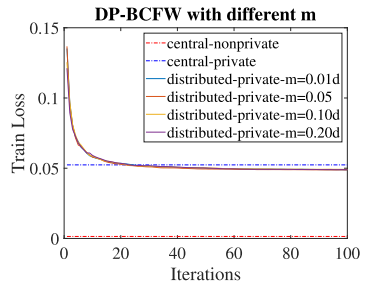

Fig. 8. Effect of changing the sketched active feature length m.

We use a series of random subsets of samples with sizes $n = [500, 1000, 2000, 4000, 8000]$ from $\boldsymbol{A}$ for private training. We then measure the excess empirical risk by taking differences of the training losses, which are between the one measured at the output of the private algorithm and the one measured at the ground truth variable. We plot the excess empirical risk (y-axis) in log against the number of training data samples (x-axis) in log (i.e., log–log scale). In the figure, we compare the centralized DP-FW [31] with our distributed BCFW with different number of user nodes ($K = 2, 4, 8, 16$). According to Theorem 4, under log–log scale plots, we expect that the figure will be approximately a decreasing linear line, because the excess empirical risk will drop given the increasing number of training samples. In addition, because of the other constants hidden by $O(\cdot)$, it will not be a strictly straight line.

The result is in Fig. 4. All algorithms have decreased privacy risk with increasing number of training data samples. In addition, with more worker nodes, the excess empirical risk gets smaller, which shows the advantage of distributed DP-BCFW we proposed. Finally, the excess empirical risk under a larger n drops a bit slower and make it not a strictly straight line, which is also consistent with our expectation.

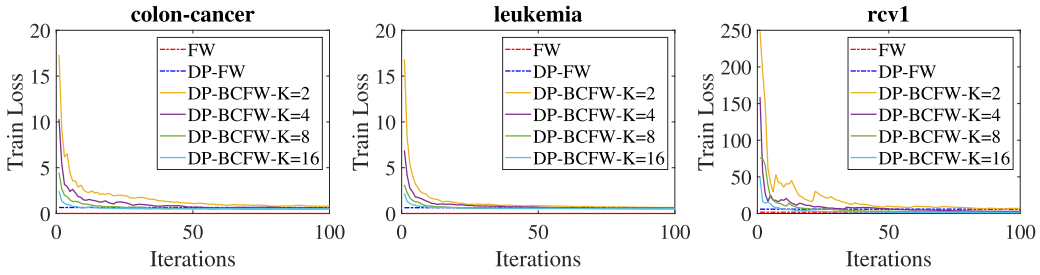

Fig. 9. Comparisons on three real datasets: colon-cancer, leukemia and rcv1.

E. Effect of Changing Parameters of DP-BCFW

We study the parameter sensitivity of our proposed algorithm. In all algorithms, we use the converged training loss of the centralized nonprivate FW and private FW [31] as baselines, while varying the parameters of our DP-BCFW algorithm. We use the same $[n, d] = [800, 8000]$ dataset with sparsity 1%.

1) *The Effect of Changing K:* For the user node K, we change it from 2, 4, 8, and 16. According to Fig. 5, all private algorithms have decreased losses compared to the nonprivate FW because of the injected noise for privacy protection purpose. More worker nodes result to better performance, which is consistent with its nonprivate counterpart.
2) *The Effect of Different ϵ:* First, we change privacy budget ϵ from 0.001 to 10 and plot the training loss versus iteration number. According to Fig. 6, the decrease of both losses are almost unaffected by the change of ϵ, which means the estimation of the gradient and active index in each iteration are still accurate enough despite the perturbations injected for privacy protection. It also supports Assumption 2 and Remark 2 that the assumption of inexactness is reasonable and the approximation errors do not affect the convergence of BCFW. Second, we fix the privacy budget to be $(\epsilon, \delta) = (1.0, 1/n)$ and vary the ratio of the privacy budget assigned to the private sign index and the private active feature to be $1:9, 3:7, 5:5, 7:3, 9:1$. The result is in Fig. 7, which shows the different division of the privacy budget ratio also does not affect the convergence.
3) *The Effect of Changing m:* According to Fig. 8, the performance of the algorithm is almost unaffected by changing of the sketched size m. Also, we can reduce the size of the communicated features to be as small as 1% of its original length, meaning the communication cost can be greatly reduced without sacrificing the performance.

F. Performance of DP-BCFW on Real Datasets

Finally, we compare the distributed DP-BCFW on three real datasets. We distribute the features on 2, 4, 8, 16 user nodes. We use the nonprivate centralized FW and the centralized DP-FW [31]. The result is in Fig. 9, which shows the same trend with the results on the synthetic datasets. That is, more user nodes result to the better performance, and the utility drop is within a reasonable range when compared to the nonprivate FW.

VI. Conclusion

In this article, we studied the featurewise distributed sparse optimization with uplink communication efficiency and DP restrictions. We approached the problem in a principled way. First, we propose the BCFW-AS and established its convergence for both exact and inexact per-iteration updates. To achieve this, we have designed a new universal step-size and defined a new expected curvature notion, both are superior than existing case-by-case crafted quantities under corresponding samplings. Equipped with the powerful inexact BCFW-AS, we have developed the differentially private BCFW-AS algorithm, which enjoys lower uplink communication and nearly optimal in utility. These advantages have been rigorously proved in theory. We have also verified the theoretical advantages by empirical evidences.

In the future, we will extend the distributed setting considered in this article to a completely decentralized setting, which eliminates the need of introducing the centralized server. We expect the decentralized extension to bring about several advantages, including: 1) more practicality in terms of system implementation because the "honest but curious" assumption for the centralized server can be challenging to ensure in practice; 2) more flexibility in terms of communication network topology where each user node can have their own set of neighbors to communicate; and 3) better scalability with respect to the number of users because it eliminates the bottleneck issue in the distributed setting where the server has increasing difficulty in handling all the uplink communication from a growing number of users.

References

[1] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Stat. Soc., Ser. B, Methodol.*, vol. 58, no. 1, pp. 267–288, Jan. 1996.
[2] V. N. Vapnik, "An overview of statistical learning theory," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 988–999, Sep. 1999.
[3] M. Tan, I. W. Tsang, and L. Wang, "Minimax sparse logistic regression for very high-dimensional feature selection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 10, pp. 1609–1622, Oct. 2013.
[4] J. Wu and H. Yang, "Linear regression-based efficient SVM learning for large-scale classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 10, pp. 2357–2369, Oct. 2015.
[5] S. Yang, Q. Liu, and J. Wang, "A collaborative neurodynamic approach to multiple-objective distributed optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 4, pp. 981–992, Apr. 2018.
[6] H. Li, Q. Zhang, J. Deng, and Z.-B. Xu, "A preference-based multi-objective evolutionary approach for sparse optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 5, pp. 1716–1731, May 2018.
[7] R. Wang, N. Xiu, and C. Zhang, "Greedy projected gradient-Newton method for sparse logistic regression," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 2, pp. 527–538, Feb. 2020.

[8] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *J. Mach. Learn. Res.*, vol. 12, pp. 1069–1109, Mar. 2011.
[9] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, 2014.
[10] R. Bassily, V. Feldman, K. Talwar, and A. G. Thakurta, "Private stochastic convex optimization with optimal rates," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 11282–11291.
[11] D. Kifer, A. Smith, and A. Thakurta, "Private convex empirical risk minimization and high-dimensional regression," *J. Mach. Learn. Res.*, to be published.
[12] R. Bassily, A. Smith, and A. Thakurta, "Private empirical risk minimization: Efficient algorithms and tight error bounds," in *Proc. IEEE 55th Annu. Symp. Found. Comput. Sci.*, Oct. 2014, pp. 464–473.
[13] R. Iyengar, J. P. Near, D. Song, O. Thakkar, A. Thakurta, and L. Wang, "Towards practical differentially private convex optimization," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 299–316.
[14] S. P. Kasiviswanathan and H. Jin, "Efficient private empirical risk minimization for high-dimensional learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 488–497.
[15] N. Agarwal, A. T. Suresh, F. X. X. Yu, S. Kumar, and B. McMahan, "cpsgd: Communication-efficient and differentially-private distributed SGD," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 7564–7575.
[16] Y. Lou, L. Yu, S. Wang, and P. Yi, "Privacy preservation in distributed subgradient optimization algorithms," *IEEE Trans. Cybern.*, vol. 48, no. 7, pp. 2154–2165, Jul. 2018.
[17] R. Jin, Y. Huang, X. He, T. Wu, and H. Dai, "Stochastic-sign SGD for federated learning with theoretical guarantees," 2020, *arXiv:2002.10940*. [Online]. Available: http://arxiv.org/abs/2002.10940
[18] S. Han, U. Topcu, and G. J. Pappas, "Differentially private distributed constrained optimization," *IEEE Trans. Autom. Control*, vol. 62, no. 1, pp. 50–64, Jan. 2017.
[19] A. T. Suresh, X. Y. Felix, S. Kumar, and H. B. McMahan, "Distributed mean estimation with limited communication," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3329–3337.
[20] W. Wen *et al.*, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1509–1519.
[21] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "signSGD: Compressed optimisation for non-convex problems," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 560–569.
[22] S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi, "Error feedback fixes signsgd and other gradient compression schemes," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3252–3261.
[23] S. Zheng, Z. Huang, and J. Kwok, "Communication-efficient distributed blockwise momentum SGD with error-feedback," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 11450–11460.
[24] D. Basu, D. Data, C. Karakus, and S. Diggavi, "Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 14695–14706.
[25] J. Que, X. Jiang, and L. Ohno-Machado, "A collaborative framework for distributed privacy-preserving support vector machine learning," in *Proc. AMIA Annu. Symp.*, 2012, p. 1350.
[26] Y. Li, X. Jiang, S. Wang, H. Xiong, and L. Ohno-Machado, "Vertical grid logistic regression (VERTIGO)," *J. Amer. Med. Inform. Assoc.*, vol. 23, no. 3, pp. 570–579, May 2016.
[27] S. Wang *et al.*, "Genome privacy: Challenges, technical approaches to mitigate risk, and ethical considerations in the united states," *Ann. New York Acad. Sci.*, vol. 1387, no. 1, pp. 73–83, Jan. 2017.
[28] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. Theertha Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv:1610.05492*. [Online]. Available: http://arxiv.org/abs/1610.05492
[29] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol. (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.
[30] P. Kairouz *et al.*, "Advances and open problems in federated learning," 2019, *arXiv:1912.04977*. [Online]. Available: http://arxiv.org/abs/1912.04977
[31] K. Talwar, A. Thakurta, and L. Zhang, "Nearly optimal private lasso," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3025–3033.
[32] A. Bellet, Y. Liang, A. B. Garakani, M.-F. Balcan, and F. Sha, "A distributed frank-wolfe algorithm for communication-efficient sparse learning," in *Proc. SIAM Int. Conf. Data Mining*, Jun. 2015, pp. 478–486.
[33] Y.-X. Wang, V. Sadhanala, W. Dai, W. Neiswanger, S. Sra, and E. Xing, "Parallel and distributed block-coordinate frank-wolfe algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1548–1557.
[34] S. Lacoste-julien, M. Jaggi, M. Schmidt, and P. Pletscher, "Block-coordinate frank-wolfe optimization for structural SVMs," in *Int. Conf. Mach. Learn.*, 2013, pp. 53–61.
[35] P. Richtárik and M. Takáč, "Parallel coordinate descent methods for big data optimization," *Math. Program.*, vol. 156, nos. 1–2, pp. 433–484, Mar. 2016.
[36] P. Richtárik and M. Takác, "Distributed coordinate descent method for learning with big data," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2657–2681, 2016.
[37] Z. Qu and P. Richtárik, "Coordinate descent with arbitrary sampling I: Algorithms and complexity," *Optim. Methods Softw.*, vol. 31, no. 5, pp. 829–857, Sep. 2016.
[38] Z. Qu and P. Richtárik, "Coordinate descent with arbitrary sampling II: Expected separable overapproximation," *Optim. Methods Softw.*, vol. 31, no. 5, pp. 858–884, Sep. 2016.
[39] J. Lou and Y. Cheung, "Uplink communication efficient differentially private sparse optimization with feature-wise distributed data," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 1–9.
[40] M. Jaggi, "Revisiting frank-wolfe: Projection-free sparse convex optimization," in *Proc. Int. Conf. Mach. Learn.*, 2013. pp. 427–435.
[41] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papailiopoulos, and S. Wright, "Atomo: Communication-efficient learning via atomic sparsification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 9850–9861.
[42] V. Gandikota, D. Kane, R. Kumar Maity, and A. Mazumdar, "VqSGD: Vector quantized stochastic gradient descent," 2019, *arXiv:1911.07971*. [Online]. Available: http://arxiv.org/abs/1911.07971
[43] N. Ivkin *et al.*, "Communication-efficient distributed SGD with sketching," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 13144–13154.
[44] Z. Liu, T. Li, V. Smith, and V. Sekar, "Enhancing the privacy of federated learning with sketching," 2019, *arXiv:1911.01812*. [Online]. Available: http://arxiv.org/abs/1911.01812
[45] C. Heinze-Deml, B. McWilliams, and N. Meinshausen, "Preserving privacy between features in distributed estimation," *Stat*, vol. 7, no. 1, p. e189, 2018.
[46] M. Zhang, L. Chen, A. Mokhtari, H. Hassani, and A. Karbasi, "Quantized frank-wolfe: Faster optimization, lower communication, and projection free," 2019, *arXiv:1902.06332*. [Online]. Available: http://arxiv.org/abs/1902.06332
[47] K. Kenthapadi, A. Korolova, I. Mironov, and N. Mishra, "Privacy via the johnson-lindenstrauss transform," *J. Privacy Confidentiality*, vol. 5, no. 1, pp. 39–71, Aug. 2013.
[48] Y. Yu, X. Zhang, and D. Schuurmans, "Generalized conditional gradient for sparse estimation," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 5279–5324, 2017.
[49] F. Hanzely and P. Richtarik, "Accelerated coordinate descent with arbitrary sampling and best rates for minibatches," in *Proc. 22nd Int. Conf. Artif. Intell. Statist.*, Apr. 2019, pp. 304–312.
[50] D. P. Woodruff *et al.*, "Sketching as a tool for numerical linear algebra," *Found. Trends Theor. Comput. Sci.*, vol. 10, nos. 1–2, pp. 1–157, 2014.
[51] *Libsvm*. Accessed: 2019. [Online]. Available: www.csie.ntu.edu.tw/~cjlin/libsvmtools

Jian Lou received the Ph.D. degree from the Department of Computer Science, Hong Kong Baptist University, Hong Kong, in 2018.

His research interests include statistical learning and numerical optimization.

Yiu-ming Cheung (Fellow, IEEE) received the Ph.D. degree from the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, in 2000.

He is currently a Full Professor with the Department of Computer Science, Hong Kong Baptist University, Hong Kong. His current research interests include machine learning, pattern recognition, visual computing, and optimization. His details can be found at: http://www.comp.hkbu.edu.hk/~ymc.