Communication Efficient and Differentially Private Logistic Regression under the Distributed Setting

Ergute Bao*
National University of Singapore
ergute@comp.nus.edu.sg

Dawei Gao
Alibaba Group
gaodawei.gdw@alibaba-inc.com

Xiaokui Xiao
National University of Singapore
xkxiao@nus.edu.sg

Yaliang Li
Alibaba Group
yaliang.li@alibaba-inc.com

ABSTRACT

We study the classic machine learning problem of logistic regression with differential privacy (DP), under the distributed setting. While logistic regression with DP has been extensively studied in the literature, most of the research is focused on the centralized setting, where a centralized server is trusted with the entire private training dataset. However, in many real-world scenarios (e.g., federated learning), the data is distributed among multiple clients who may not trust others, including clients and the server. While the server tries to learn a model using the clients' private datasets, the clients should provide each individual record in their local datasets with a formal privacy guarantee.

Towards this end, we propose a general mechanism for logistic regression with DP under the distributed setting, based on output perturbation. We show that our solution satisfies differential privacy and enjoys privacy amplification by secure aggregation, a recent technique for DP under the distributed setting. In addition, our solution also incurs much lower communication costs, compared with existing ones. In particular, our solution requires the clients to communicate only once throughout the entire FL process. Finally, we provide experimental results on real-world datasets to demonstrate the effectiveness of our solution.

CCS CONCEPTS

• **Security and privacy → Data anonymization and sanitization**.

KEYWORDS

Privacy, Differential Privacy, Federated Learning

ACM Reference Format:
Ergute Bao, Dawei Gao, Xiaokui Xiao, and Yaliang Li. 2023. Communication Efficient and Differentially Private Logistic Regression under the Distributed Setting. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3580305.3599279

*Part of this work was done when the first author was an intern at Alibaba Group.

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08.
https://doi.org/10.1145/3580305.3599279

1 INTRODUCTION

Differential Privacy (DP) [27] is the state-of-the-art algorithmic framework for preserving an individual level of privacy. DP has been widely studied in the academia [8, 29, 30, 33, 34, 54], and deployed in the industry [5, 23, 31, 55]. The classic centralized DP framework assumes a trusted server, who runs a randomized mechanism on an input private dataset, and outputs perturbed statistics (e.g., [7, 25, 27, 28]), a sanitized dataset (e.g., [34, 48, 66]), a randomized decision (e.g., [47, 52]), or perturbed parameters for a machine learning model (e.g., [1, 18, 62]), etc. The canonical way to turn a non-DP mechanism into its differentially private version is to let the server inject random noise, whose scale matches the influence of an individual record from the input dataset, to the output of the underlying non-DP mechanism. Intuitively, this means that on observing the output, any adversary cannot infer whether any given individual's record is included in the input dataset or not, since the individual's influence is already masked by the noise. With that being said, individual privacy is achieved.

Logistic Regression (LR) [22, 46] is a classic algorithm in Machine Learning. In the standard setting of LR, there are two classes of data, encoded by a binary variable in $\{-1, 1\}$. The goal of LR is to find a model that correctly predicts the value of the binary variable, based on a given data point. For example, given a patient's record, an LR model may predict whether the patient has diabetes or not, using the model parameters. Logistic regression with DP has been extensively studied in the literature. Proposed methods include objective perturbation [18], output perturbation [19] and gradient perturbation [1, 62]. Note that all these methods focus on the centralized setting where a trusted server holds the entire private training dataset and runs a centralized-DP mechanism on the dataset.

In real-world scenarios, however, the private data is sometimes distributed among multiple clients who do not trust each other. In the meantime, an un-trusted server wants to learn a global model using the clients' private datasets. The goal of these clients is to collaboratively learn a global model without sharing their private data. This setting emerges from Federated Learning (FL) [11, 16, 17, 41, 44, 51, 61]. Note that it is more challenging to achieve DP under the distributed setting while maintaining the same level of utility as in the centralized setting, where the server is trustworthy. Intuitively, this is because each client must independently inject a sufficient amount of random noise on her side before sharing any information (e.g., gradients, model parameters, etc.) with others, to achieve an individual level of privacy for her local dataset. In the

meantime, the independent random noises from different clients accumulate into a much larger random noise, and ultimately cause utility degradation for the global model.

The recent techniques of Secure Aggregation [12] and integer-valued DP noise [3, 4, 40] together produce an elegant solution to improve the privacy-utility trade-off for the distributed setting. To be more specific, [3, 4, 40] show that, the global model trained using gradient perturbation under the distributed setting achieves comparable utility as the one trained under the centralized setting, while maintaining the same privacy guarantee for individual records. The key idea is as follows. The clients first independently inject integer-valued random noise into their discretized gradients. After that, the clients collectively participate in the Secure Aggregation protocol [12] (SecAgg) to securely compute the sum of their perturbed gradients. Finally, the server retrieves the aggregate result from the clients. Note that the only information that the server learns about the private gradients is the aggregate result, due to the security property of SecAgg. In addition, since independent integer-valued noises from clients aggregate into noise with a larger variance, the overall variance of the noise in the aggregate output is N times larger than each individual noise on the client side (N is the number of clients). Hence, from the server's perspective, the gradient sum is effectively perturbed by the aggregation of all clients' noises. As a result, the privacy guarantee is amplified by a factor of $1/\sqrt{N}$, matching that in the centralized setting.

Limitations. However, the above-mentioned gradient perturbation technique, which is based on gradient descent and its variants, incurs high communication costs when used under the distributed setting [10, 12, 38]. To be more specific, at every round, all (or a subset of) clients need to share their gradients with the server through the SecAgg protocol, and wait for the server to broadcast the updated model parameters. This process is repeated thousands number of times until the FL process ends. From a practical perspective, a larger number of communications rounds also possibly lead to larger privacy risks, such as eavesdropping. For example, an adversary could have a higher chance of successfully eavesdropping on the communication channel between clients. Once the adversary succeeds, the privacy guarantee of the whole DP training mechanism would collapse, as it is based on the assumption of secure and reliable communication channels in the SecAgg protocol as well as in the general secure multiparty computation (MPC) models.

In addition, as shown in [19], gradient perturbation algorithms incur higher privacy costs for LR with DP while maintaining the same model utility as output perturbation algorithms, under the centralized setting, due to repeatedly computing sensitive information over the same input dataset. The high-level explanation is that the original analysis for DPSGD does not take the smoothness and strong convexity of the loss function of logistic regression. Instead, Chen et al. [19] show that the final model parameters of RSGD-AR have bounded sensitivity by utilizing such properties. Hence, it is reasonable to suspect that gradient perturbation may also incur high privacy costs under the distributed setting as well. Both points regarding privacy overheads and communication overheads suggest we develop a new solution for logistic regression with DP under the distributed setting that achieves better privacy-utility trade-off and incurs low communication costs simultaneously.

Our contributions. In this paper, we propose a mechanism for logistic regression with differential privacy under distributed setting, named Sk-RSGD-AR. As we have mentioned earlier, gradient perturbation may not be the best choice under the distributed setting. Instead, we use the state-of-the-art output perturbation mechanism for LR under centralized DP (i.e., RSGD-AR [19]) as the building block for our solution. We also incorporate integer-valued DP noise under the distributed setting into our solution. Through careful privacy reasoning, we show that Sk-RSGD-AR achieves a formal privacy guarantee and in the meantime, incurs significantly lower communication costs than gradient perturbation mechanisms. In particular, all clients need to participate in SecAgg only once to share their aggregate perturbed model to the server at the very end of the training process. Finally, we conduct experiments on real-world datasets to demonstrate the effectiveness of our solution.

2 PRELIMINARIES

2.1 Differential Privacy

Differentially Privacy (DP) ensures individual level privacy by requiring that the output distributions of a randomized mechanism $\mathcal{M}$ are similar for neighboring input datasets D and D', where two datasets D and D' are called neighboring (written as $D \sim D'$) if one can be obtained from the other by replacing one individual record. Intuitively, this means that from the outcome of $\mathcal{M}$, one can not infer whether any particular individual participates in the input dataset or not. As a result, the individual level of privacy is achieved for records in the dataset. The standard definition of differential privacy is (ϵ, δ)-DP [27].

Definition 1 ((ϵ, δ)-Differential Privacy [27]). *Consider positive parameters ϵ and δ, we say a mechanism $\mathcal{M}$ satisfies (ϵ, δ)-differential privacy (DP) if the following holds for any set of output $O \subseteq Range(\mathcal{M})$ and any neighboring datasets D and D'.*

$$\Pr[\mathcal{M}(D) \in O] \leq \exp(\epsilon) \cdot \Pr[\mathcal{M}(D') \in O] + \delta.$$

The parameters ϵ and δ in Definition 1 represent the level of privacy protection. Larger values of ϵ and δ imply weaker level of privacy guarantees and smaller values of ϵ and δ imply stronger level of privacy guarantees. Intuitively, this is because as ϵ and δ decrease, it becomes more difficult for an adversary to distinguish the outcome from distributions $\mathcal{M}(D)$ and $\mathcal{M}(D')$. As we can see, (ϵ, δ)-DP characterizes the worst-case privacy guarantee for a mechanism, considering all possible outputs.

An alternative definition called Rényi-Differential Privacy (RDP) [54] is based on the concept of Rényi Divergence, reviewed next.

Definition 2 (Rényi Divergence [63]). *Assuming that distributions P and Q are defined over the same domain, and P is absolute continuous with respect to Q, then the Rényi divergence of P from Q of finite order $\alpha \in (0, 1) \cup (1, \infty)$ is defined as:*

$$D_\alpha(P \,\|\, Q) = \frac{1}{\alpha - 1} \log \mathbb{E}_{X \sim P} \left[\left(\frac{P(X)}{Q(X)} \right)^{\alpha - 1} \right].$$

Definition 3 (Rényi Differential Privacy (RDP) [54]). *Consider positive parameters α ($\alpha \neq 1$) and ϵ, we say a mechanism $\mathcal{M}$ satisfies (α, ϵ)-Rényi differential privacy (RDP) if for all neighboring*

datasets D and D', the following holds

$$D_\alpha(\mathcal{M}(D) \,\|\, \mathcal{M}(D')) \leq \epsilon.$$

To ensure the differential privacy of a base mechanism, it suffices to inject random noise into its outcome. The scale of the noise should be proportional to the sensitivity of the base mechanism [27], defined as follows.

Definition 4 (Sensitivity). *The sensitivity of a d-dimensional function g which takes input from domain $\mathcal{D}$, denoted as $S(g)$, is defined as follows*

$$S(g) = \max_{D \sim D'} \|g(D) - g(D')\|,$$

where $\|\cdot\|$ is a norm. Here the maximum is taken over all possible neighboring datasets D and D' that belongs to the input domain $\mathcal{D}$.

For example, injecting continuous Gaussian noise sampled from $\mathcal{N}(0, \sigma^2)$ to each dimension of function g satisfies $(\alpha, \alpha S^2(g)/(2\sigma^2))$-RDP [54]. Here $S(g)$ stands for the $\mathcal{L}_2$ sensitivity of function g.

Finally, given any $0 < \delta < 1$, we can convert the privacy protection of a randomized mechanism under (α, ϵ)-RDP to that under (ϵ', δ)-DP using the following conversion rule.

Lemma 1 (Conversion rule for (α, τ)-RDP [15]). *For any $\alpha \in (1, \infty)$, if $D_\alpha(\mathcal{M}(D) \,\|\, \mathcal{M}(D')) \leq \epsilon$ for any neighboring datasets D and D', then $\mathcal{M}$ satisfies (ϵ', δ)-DP for*

$$\epsilon' = \epsilon + \frac{\log(1/\delta) + (\alpha - 1)\log(1 - 1/\alpha) - \log(\alpha)}{\alpha - 1}.$$

2.2 DP under Distributed Settings

Recall that the classic centralized differential privacy framework [27] assumes that a single server is trusted with the entire private dataset whereas in distributed settings, such as Federated Learning [44, 51], the private dataset is partitioned to different clients, who do not trust other clients and the server. In this distributed setting, the server tries to learn a function of interest calculated on the entire private dataset (e.g., the population mean), while the clients want to maintain an individual level of privacy for their local private datasets.

Distributed Differential Privacy [4, 40] suggest the following solution. Each client first independently perturbs the outcome of the function of interest calculated on her local private dataset (e.g., gradients computed on local data) with some random noise. Next, the clients use SecAgg [12], a secure multiparty computation protocol, to securely compute the sum of the perturbed outcomes from all participating clients. The server then retrieves the aggregate result. Note that since SecAgg is cryptographically secure, the only information that the server learns is the aggregate result. Namely, from the server's perspective, the aggregate result is effectively perturbed by the sum of the individual random noises injected on the client side. The differential privacy guarantee of such a mechanism is determined by the sensitivity of the function of interest and the *overall* variance of the aggregate random noise contributed by all clients, instead of the random noise from of a single client. Roughly speaking, the privacy level for individual records from the client's local dataset is amplified by a factor of $1/\sqrt{N}$. Intuitively, such privacy amplification is done by hiding a client's identity among other clients, since all the server observes is the aggregate outcome. We refer interested readers to [4, 30, 40] for more detailed discussions on distributed DP and on its privacy amplification.

Note that the original notion of Distributed-DP protects a client-level of privacy [3, 4, 40] instead of a record-level of privacy, which is the focus of this paper. We would like to note that the idea of combining secure aggregation with differentially private noise can still be used in our setting to protect a record level of DP. This is because to protect a record (resp. client), it suffices to inject random noise of scale proportional to the sensitivity of the function of interest, where the sensitivity measures the change of the function due to a record (resp. client). Note that in both settings, the entire private dataset is partitioned to N clients, and the added (or removed) private training item/example is located in exactly one of the clients. In our setting, exactly one training item from the entire private dataset contributes to the difference in the aggregate statistics. In particular, for gradient perturbation, the corresponding sensitivity is the clipping norm for the gradient of a training record/item.

Remark. Note that utilizing SecAgg alone as well as other MPC protocols do not provide a rigorous statistical privacy guarantee regarding the inputs in FL, in the sense that on observing the outcome of MPC, a curious adversary is still able to extract information about the private input datasets, as pointed out in [37, 53]. On the other hand, using DP alone for FL applications leads to unsatisfactory privacy-utility trade-offs, compared with the centralized setting [40]. Instead, when combining DP with MPC under FL, as done in Distributed-DP, we are able to achieve privacy-utility trade-offs comparable to the centralized setting, as MPC simulates the benefits of a trusted data curator in the centralized setting.

In the rest of this paper, we see SecAgg as a black-box for secure aggregation, which takes inputs from clients and returns the sum of their inputs. Following the assumption of previous works in DP FL [3, 12, 40], we focus on the semi-honest setting, i.e., all clients and the server strictly follow the noise injection and MPC protocol while trying to infer other clients' private inputs.

2.3 Skellam Noise

SecAgg [12] is not directly compatible with the traditional real-valued random noise such as Gaussian and Laplace noises [7, 27, 28]. This is because SecAgg is based on MPC, which only operates over finite integers. Accordingly, to achieve privacy, the client should inject an integer-valued random noise into an integer-valued function. While one can easily turn a general real-valued function into its integer-valued version by stochastically rounding the function's output to its nearest integers [4, 40] (we will see in Section 4), integer-valued noises are more difficult to design, implement and analyze.

Agarwal et al. [4] propose to use binomial noise instead of continuous Gaussian noise [28]. However, the binomial noise has a heavy tail, which leads to unsatisfactory privacy analysis under common settings in FL. To address this issue [15, 40] propose to use the discrete Gaussian noise and [3] propose the symmetric Skellam noise. Compared with discrete Gaussian noise, symmetric Skellam noise is easier to implement on digital computers, as it is the difference between two identical and independent Poisson variables. We next review the Skellam noise in more detail.

Let independent random variates Y_1 and Y_2 follow the same Poisson distribution of parameter λ. Then we say a random variate Z follows a symmetric Skellam distribution (abbreviated as Skellam distribution), if it is the difference between Y_1 and Y_2. We write $Z \sim \mathrm{Sk}(\lambda)$. We can derive the probability distribution of Z as

$$\Pr[Z = k] = \exp(-2\lambda) I_{|k|}(2\lambda), k = 0, \pm 1, \pm 2, \ldots,$$

where $I_v(u) \triangleq \sum_{h=0}^{\infty} \frac{1}{h!\Gamma(h+v+1)} \left(\frac{u}{2}\right)^{2h+v}$ is called the modified Bessel function of the first kind. By linearity of expectation, Z has mean 0 and variance 2λ. We also note that Skellam distributions have a favorable *aggregatable* property for privacy amplification using SecAgg under the distributed setting: the sum of any two independent Skellam variates is still a Skellam variate. This is because the sum of two independent Poisson variates is still a Poisson variate. In particular, when N clients all independently inject identical noises sampled from $\mathrm{Sk}(\lambda)$ to their private data (e.g., gradients) and send the perturbed data as input to SecAgg, the privacy guarantee of the output by SecAgg is effectively determined by the aggregate Skellam noise $\mathrm{Sk}(N\lambda)$, which has variance $2N\lambda$.

In the rest of this paper, we use $Z \sim \mathrm{Sk}^d(\lambda)$ to denote a d-dimensional random variable $Z \in \mathbb{Z}^d$, where each dimension of Z is independently sampled from Skellam distribution $\mathrm{Sk}(\lambda)$. In addition,Ref. [3] show that Skellam noise achieves RDP.

Lemma 2. *Let Δ_1 and Δ_2 be the $\mathcal{L}_1$ and $\mathcal{L}_2$ sensitivities of an integer-valued function g, then independently injecting Skellam noise sampled from $\mathrm{Sk}(\lambda)$ to all dimensions of g satisfies (α, ϵ)-RDP for $\alpha \in \mathbb{Z}, \alpha > 1$, and*

$$\epsilon = \frac{\alpha \Delta_2^2}{4\lambda} + \min\left(\frac{(2\alpha - 1)\Delta_2^2 + 6\Delta_1}{16\lambda^2}, \frac{3\Delta_1}{4\lambda}\right).$$

3 PROBLEM STATEMENT

3.1 Logistic Regression with Differential Privacy

Our goal is to design a privacy preserving mechanism $\mathcal{M}$ that solves the following optimization problem of $\mathbf{w}$, written as follows.

$$\arg\min_{\mathbf{w} \in \Theta} F(\mathbf{w}) = \arg\min_{\mathbf{w} \in \Theta} \frac{1}{|S|} \sum_{(\mathbf{x}, y) \in S} f(\mathbf{w}, (\mathbf{x}, y)). \tag{1}$$

where S is the input dataset to mechanism $\mathcal{M}$. For each record $(\mathbf{x}, y) \in S$, the vector $\mathbf{x} \in \mathbb{R}^d$ denotes the feature vector, and the binary value $y \in \{-1, 1\}$ represents the label. Here function f represents the loss per record, and Θ is a convex space for the model parameters. We write $\langle \mathbf{u}, \mathbf{v} \rangle$ as the inner product (element-wise product) of two vectors $\mathbf{u}$ and $\mathbf{v}$. Following previous work [19], we define the loss function of logistic regression as follows.

$$f(\mathbf{w}, \mathbf{x}, y) = \log(1 + \exp(-y \cdot \langle \mathbf{w}, \mathbf{x} \rangle)) + \frac{\mu}{2} \|\mathbf{w}\|_2^2. \tag{2}$$

Without loss of generality, we assume all feature vectors have a unit norm (i.e., $\|\mathbf{x}\|_2 = 1$), since otherwise, we can normalize the features. It can be easily shown that function $f(\mathbf{w}, \mathbf{x}, y)$ is $(1/4 + \mu)$-smooth and μ-strongly convex with respect to its first argument. In our experiments, we fix μ to 0.001, following previous work [19].

Under the distributed setting, the private dataset S is partitioned across N clients. Namely, each client i holds a partition S_i of the entire dataset, and $\cup_i S_i = S$, $S_i \cap S_j = \emptyset$ when $i \neq j$. For simplicity, we assume that the size of partition $|S_i|$ is equal for all clients $i \in [N]$. In terms of privacy, we require that $\mathcal{M}$'s outcome distribution should be similar regardless of the participation of any individual record in the input private dataset S. Namely, $\mathcal{M}$ satisfies DP. In particular, we consider the Rényi Differential Privacy framework. Formally, for some $\alpha > 1$ and ϵ, and any neighboring private training datasets S and S' differing by one record, the output distribution of $\mathcal{M}$ satisfies $D_\alpha(\mathcal{M}(S) \| \mathcal{M}(S')) \leq \epsilon$.

3.2 Motivation

The well-known approach to solve the optimization problem defined as in Eq. (1) under the centralized setting is through gradient perturbation [1, 62], i.e., DPSGD. DPSGD is built upon the classic stochastic gradient descent (i.e., SGD [14, 42, 59]) with two modifications for privacy: clipping and noise injection. To be more specific, at every iteration, the server first samples a subset of the input private dataset; computes and clips each individual gradient from the batch; injects Gaussian noise to the gradient sum; and updates the model parameters using the noisy sum. Here the scale of the random noise is roughly $O(c\sqrt{T})$, where c is the clipping norm for the gradient and T is the overall number of iterations. Naively running DPSGD on the client side and letting the server update the global model using the aggregate value of the noisy gradient sums broadcast by the clients incurs a noise overhead of $\sqrt{N}$ (N is the number of clients), compared to the centralized setting. Integer-valued DP noise [3, 4, 40] with SecAgg [12] closes this gap by exploiting the security property of SecAgg, as we have mentioned in Section 2.2. However, gradient perturbation under the distributed setting incurs excessive communication costs due to frequent gradient sharing among the clients and the server, especially for edge devices and MPC protocols. In addition, from a practical perspective, a larger number of communications rounds also possibly lead to larger privacy risks, such as eavesdropping, as we have mentioned in Section 1. In this paper, we address the issue of communication costs for learning a DP LR model under the distributed setting.

Chen et al. [19] propose RSGD-AR to solve LR with DP under the centralized setting. The underlying learning algorithm of RSGD-AR is close to the classic mini-batch gradient descent algorithm [14, 42, 59]. Their paper show that, the final model parameters of RSGD-AR have bounded sensitivity through careful analysis that exploits the smoothness and strong convexity of the loss function of LR. Hence, directly injecting random noise to the parameters guarantees DP, as shown in [19]. We refer interested readers to [9, 19, 32, 64] for further details on how to utilize the smoothness and strong convexity conditions for privacy analysis.

Based on RSGD-AR, we propose a communication-efficient solution for learning a private logistic regression model under the distributed setting. The idea is to let each client independently run the non-private version of RSGD-AR on her local data partition, and use Skellam noise [3] to perturb her local model and use SecAgg [12] to aggregate the perturbed models. Similar to previous approaches [3, 4, 40], SecAgg amplifies the privacy guarantee for each private training data item. In the meantime, compared with gradient perturbation under the distributed setting, our solution

Algorithm 1: Sk-RSGD-AR

Input: Initial model weights $w \in \mathbb{R}^d$; input dataset $S = (S_1, \ldots, S_N)$; number of local update epochs T_l; local step size η_l; number of local mini-batches m; scale parameter γ; bias parameter β; noise parameter λ.

1 The server shares the initial model weights w to all clients.
2 **for** $i \in 1 \ldots N$ **do**
3 $\quad w_i \leftarrow$ Algorithm 2$(S_i, w, T_l, \eta_l, m, \tau)$. // local update
4 $\quad v_i \leftarrow w_i - w$. // compute the local model change.
5 $\quad v_i^* \leftarrow$ Algorithm 3(v_i, γ, β). // discretization.
6 $\quad \tilde{v}_i \leftarrow v_i^* + \mathrm{Sk}^d(\lambda)$. // perturbation.
7 $\bar{v} \leftarrow \mathrm{SecAgg}(\{\tilde{v}_i\}_{i \in [N]})$. // secure aggregation
8 $\bar{v} \leftarrow \frac{1}{\gamma} \cdot \bar{v}$. // sum retrieval by the server
9 $w \leftarrow w + \bar{v}/N$. // model update

Output: w model learnt on S.

Algorithm 2: Client Procedure for Local Update

Input: Input dataset $S_i = ((x_1, y_1), \ldots, (x_n, y_n))$; initial model weights w_0; number of epochs T; initial step size η_0; number of mini-batches m; averaging interval τ.

1 Randomly permute the dataset S_i.
2 Construct mini-batches $B_1, \ldots, B_m$ of equal size $|B|$.
3 $t \leftarrow 0, h \leftarrow 0$.
4 **for** $s \in 1 \ldots T$ **do**
5 $\quad h \leftarrow h + 1$.
6 $\quad \eta \leftarrow \eta_0 / h$.
7 $\quad$ **for** $j \in 1 \ldots m$ **do**
8 $\quad\quad t \leftarrow t + 1$.
9 $\quad\quad g \leftarrow \frac{1}{|B_j|} \sum_{i \in B_j} \nabla f(w, x_i)$.
10 $\quad\quad w_t \leftarrow w_{t-1} - \eta g$
11 $\quad$ **if** $s \mod \tau = 0$ **then**
12 $\quad\quad w_t \leftarrow \frac{1}{m\tau} \sum_{k=0}^{m\tau - 1} w_{t-k}$. // Averaging
13 $\quad\quad h \leftarrow 0$. // step size reset
14 $\quad\quad t \leftarrow t + 1$

Output: w model learnt on S_i.

Algorithm 3: Client Procedure for Discretization

Input: Local model change v_i; scale ratio γ; bias parameter β.

1 $\hat{v}_i \leftarrow \gamma \cdot v_i$. // scaling
2 **repeat**
3 $\quad$ Let v_i^* be the stochastic rounded result of $\hat{v}_i$. Namely $\mathbb{E}[v_i^*] = \hat{v}_i$, and $\|v_i^* - \hat{v}_i\|_\infty \leq 1$.
4 **until** $\|v_i^* - \hat{v}_i\|_2 \leq \beta \cdot \sqrt{d}$

Output: $v_i^* \in \mathbb{Z}^d$ for noise injection.

significantly reduces the communication cost. This is because, in our solution, the clients participate in the secure aggregation protocol only once after the end of local training processes. We call our solution Sk-RSGD-AR, introduced in detail next.

4 OUR SOLUTION

We present our solution Sk-RSGD-AR for logistic regression with differential privacy under the distributed setting, outlined as in Algorithm 1. We first give an overview of our solution. At the very beginning, the server first shares the initial model weights to all clients (Line 1 in Algorithm 1). Each client performs local model update using Algorithm 2 for T_l local epochs (Line 3 in Algorithm 1). and computes the local model change (Line 4 in Algorithm 1). After this, the client discretizes the local model change using Algorithm 3 (Line 5 in Algorithm 1), to set up the stage for integer-valued noise injection and SecAgg (explained in more details next). After discretization, each client then independently perturbs the discretized change with a d-dimensional Skellam noise $\mathrm{Sk}^d(\lambda)$ (Line 6 in Algorithm 1). After all clients have done the perturbation, they collectively participate in SecAgg to aggregate their perturbed model changes in a secure manner (Line 7 in Algorithm 1). Upon receiving the output of SecAgg, the server reconstructs the aggregate model change (Line 8 in Algorithm 1) and updates the model weights accordingly (Line 9 in Algorithm 1).

Algorithm 2 is the non-DP version of RSGD-AR in [19], and we refer interested readers to the original paper for its detailed algorithmic description. Roughly speaking, the algorithm first randomly permutes the input dataset and then partitions the dataset into m batches, and then repeatedly uses these batches for computing the gradient mean and updates the model accordingly. In addition, after every τ epochs, RSGD-AR performs an averaging over all model parameters from the previous τ epochs. Note that the random permutation step at the beginning improves over the worst-case sensitivity analysis, as mentioned in [19]. Roughly speaking, this is because the differing record between two neighboring input datasets could locate in the first batch, causing a smaller influence on the final model parameters, compared to the case when the record is located in the last batch. This step also adds technicality to our privacy analysis, as we will see in Section 4.1

Next, we explain the discretization process (see Algorithm 3) for the change of local model weights in more detail, which sets up the stage for integer-valued noise injection, as the general real-valued private information (i.e., model parameters) and real-valued noise (e.g., continuous Gaussian noise) are not directly compatible with SecAgg, which is based on secure multiparty computation (MPC). The client scales the input vector by γ. Intuitively, a larger γ means finer quantization granularity and requires larger random noise to preserve privacy (since scaling increases the $\mathcal{L}_2$ norm by a factor of γ), and vice versa.

The client then repeats the same stochastic rounding process on the scaled vector (Line 3 in Algorithm 3) until its discretized version satisfies a certain condition. Here the stochastic rounding process rounds a value in form $X + Y$, where X is the integer part and Y is the fractional part, to $X + 1$ with probability Y, and to X with probability $(1 - Y)$. For example, 1.85 is rounded to 2 with probability 0.85, and to 1 with probability 0.15. The condition specifies the $\mathcal{L}_2$ distance between the original vector and its discretized version, which restricts the sensitivity overhead due to stochastic rounding to a pre-fixed value quantified by $\beta \in (0, 1)$. This condition basically requires that at most β fraction of the values in the vector cause a sensitivity increase in their dimensions (e.g. when -1.1 is rounded to -2, or when 2.9 is rounded to 3). Intuitively, the parameter β controls the variance-bias trade-off. A smaller β means more bias, since more rounded results would not satisfy the condition, but also

smaller variance due to DP noise, since less sensitivity overhead is introduced. In particular, the conditional rounding process may never stop when $\beta = 0$, since all rounded results are rejected; and the process degenerates to the unconditional stochastic rounding process when $\beta = 1$.

Remark. We emphasize that since we use Skellam noise as the additive DP noise on the client side, Sk-RSGD-AR enjoys the nice privacy amplification property of distributed-DP (as we will see in the privacy analysis in Section 4.1). In addition, our solution Sk-RSGD-AR incurs low communication costs since all clients need only communicate once, i.e., participating in SecAgg to securely share the sum of their perturbed local model changes to the server, at the very end of the training process.

Random rotation. Note that before the clients perform the rounding step on their local sides (Line 2 in Algorithm 3) , they have the option to first randomly rotate the local model change with a random Walsh-Hadarmard transform (random rotation), which preserves the $\mathcal{L}_2$ sensitivity of the local model change (hence, privacy is not affected) while evenly distributing the signal to all dimensions [36, 40, 65]. Accordingly, the server needs to reverse such a transform after obtaining the outcome from SecAgg (Line 8 in Algorithm 1). This random rotation step is a classic trick for reducing communication bandwidth, as it evenly distributes the signal to all dimensions. But this trick does not affect the main algorithmic component of our solution, which is to reduce the number of communication rounds from thousands to only one. In our experiments, we simply fix the number of bits per dimension to 8 (which is already very small), and we have observed no significant difference in terms of model utility with/without such random rotation. We refer interested readers to [40] for a detailed discussion of randomized rotation and conditional rounding processes.

4.1 Privacy Analysis

We first present the privacy analysis of Sk-RSGD-AR when exactly one client participates in the process. To do this, we need to first reason the sensitivity of Algorithm 2 on the client side. Without loss of generality we consider any two neighboring datasets $D = \{r_1, r_2, \ldots, r_{n-1}, r_n\}$ and $D' = \{r_1, r_2, \ldots, r_{n-1}, r'_n\}$, whose differing records are r_n and r'_n. Recall that Algorithm 2 first randomly permutes the input dataset before partitioning, which is equivalent to random partitioning. Hence, when we fix the partitioning of the first $n-1$ records in D and D', the differing two records could be in any one of the m batches, with equal probability $1/m$. Accordingly, the sensitivity of the local model change (due to the differing records) also has m equal possibilities of $1/m$. In particular, the largest sensitivity corresponds to the case when the differing records are in the last batch and the smallest sensitivity corresponds to the case when they are in the first batch.

Chen et al. [19] provide a tractable way for calculating each possible $\mathcal{L}_2$ sensitivity, outlined in Algorithm 4. Note that the output of the Algorithm 4 is also an array, where the j-th entry (i.e., $\Delta[j]$, $j = 1, \ldots, m$) of the output array Δ of length m corresponds to the $\mathcal{L}_2$ sensitivity of local model change when the differing records are in the j-th batch. We refer interested readers to [19] for detailed proof of the correctness of the algorithm.

Algorithm 4: Sensitivity Computation for RSGD-AR

Input: loss function f with convexity parameter μ and smoothness parameter L; number of epochs T; initial step size η_0; number of mini-batches m; mini-batch size $|B|$.

1 **for** $j \in 1 \ldots m$ **do**
2 $\Delta[j] \leftarrow 0$. // Initialization
3 **for** $s \in 1 \ldots T$ **do**
4 $\eta \leftarrow \eta_0/s$.
5 $\rho \leftarrow \max\{|1 - \eta\mu|, |1 - \eta L|\}$.
6 **for** $j \in 1 \ldots m$ **do**
7 $\Delta[:] \leftarrow \rho\Delta[:]$. // contraction for all
8 $\Delta[j] \leftarrow \Delta[j] + \frac{2\eta R}{|B|}$. // expansion for current j

Output: Δ.

Without loss of generality we first focus on the case when the differing records are located in the first batch. Accordingly, the $\mathcal{L}_2$ sensitivity after the discretization step is written as $\gamma\Delta[1] + 2\beta \cdot \sqrt{d}$, by the fact that random Walsh-Hadamard transform preserves $\mathcal{L}_2$ norm of the vector and the triangle inequality. Formally, we denote $\mathcal{M}^{[1]}_{\text{dis}}(D)$ ($\mathcal{M}^{[1]}_{\text{dis}}(D')$, respectively) as the value of v^*_i of Algorithm 1 up to Line 5 after discretization step ($i = 1$ since there is only one client), when the n-th record r_n in D (r'_n in D', respectively) is located in the first batch. Then we have that

$$\Delta^*_2[1] \triangleq \|\mathcal{M}^{[1]}_{\text{dis}}(D) - \mathcal{M}^{[1]}_{\text{dis}}(D')\|_2 \le \gamma\Delta[1] + 2\beta\sqrt{d}.$$

The corresponding $\mathcal{L}_1$ sensitivity is bounded by

$$\Delta^*_1[1] \le \min\left(\sqrt{d}\Delta^*_2[1], (\Delta^*_2[1])^2\right).$$

Next, we study the divergence between the distributions of $\tilde{v}_i$ of Algorithm 1 up to Line 6 after noise injection of $\text{Sk}(\lambda)$, denoted as $D_\alpha(\mathcal{M}^{[1]}_{\text{Sk}(\lambda)}(D) \,\|\, \mathcal{M}^{[1]}_{\text{Sk}(\lambda)}(D'))$, when the differing n-th record r_n in D and r'_n in D' are located in the first batch. Applying Lemma 2, we have that

$$D_\alpha(\mathcal{M}^{[1]}_{\text{Sk}(\lambda)}(D) \,\|\, \mathcal{M}^{[1]}_{\text{Sk}(\lambda)}(D')) \le \epsilon_\alpha(\Delta^*_2[1], \Delta^*_1[1], \lambda),$$

where we define

$$\epsilon_\alpha(\Delta_2, \Delta_1, \lambda) \triangleq \frac{\alpha \cdot \Delta_2^2}{4\lambda} + \min\left(\frac{(2\alpha - 1)\Delta_2^2 + 6\Delta_1}{4\lambda^2}, \frac{3\Delta_1}{4\lambda}\right). \quad (3)$$

Next, we consider the general case when the differing record could be in any of the m batches. Towards that end, we first denote $\mathcal{M}^{[j]}(D)$ and $\mathcal{M}^{[j]}(D')$ as the output distributions of Algorithm 1 when the differing n-th record r_n of D and r'_n of D' are located in the j-th batch, for $j = 1, \ldots, m$. Then we can see $\mathcal{M}(D)$ as a mixture of m distributions, written as $\mathcal{M}(D) = \sum_{j=1}^m \frac{1}{m} \cdot \mathcal{M}^{[j]}(D)$. Namely, with probability $1/m$, the output distribution of $\mathcal{M}(D)$ is the same as $\mathcal{M}^{[j]}(D)$ for $j = 1, \ldots, m$. A similar expression also holds for $\mathcal{M}(D')$. Chen et al.[19] have shown that

$$D_\alpha(\mathcal{M}(S') \,\|\, \mathcal{M}(S)) \le \log\left(\sum_{j=1}^m \frac{1}{m}\mathbb{E}_{\mathcal{M}^{[j]}(S)}\left[\left(\frac{\mathcal{M}^{[j]}(S')}{\mathcal{M}^{[j]}(S)}\right)^\alpha\right]\right), \quad (4)$$

which basically means that the Rényi divergence between $\mathcal{M}(D)$ and $\mathcal{M}(D')$ is the average (taken within logarithm) of the Rényi

divergences between $\mathcal{M}^{[j]}(D)$ and $\mathcal{M}^{[j]}(D')$ for all $j = 1, \ldots, m$. We refer interested readers to the original paper of [19] for further details (in their proof of Lemma 3). Combining Eq. (4) and the fact that there is only one client and that post-processing preserves privacy, we have

$$D_\alpha(\mathcal{M}(S') \,\|\, \mathcal{M}(S)) \leq \log\left(\sum_{j=1}^{m} \frac{1}{m} \cdot \exp((\alpha-1)\cdot\epsilon_\alpha(\Delta_2^*[j], \Delta_1^*[j], \lambda))\right).$$

Next we consider the general case when there are N clients. As we have mentioned in Section 2.2, when using SecAgg [12] with Skellam noise $\mathrm{Sk}(\lambda)$ injected on each client, the aggregate local model change is effectively perturbed by $\mathrm{Sk}(N\lambda)$. In addition, since the differing record appears in exactly one of the N partitions, the above sensitivity analysis remains. Finally, we take the randomness in permuting the first $n-1$ common records of D and D' into consideration. Note that Rényi divergence is jointly quasi-convex [63], which basically means that it suffices for us to consider the worst case randomness in the permutation for the first $n-1$ common records of D and D', which also has no effect on the sensitivity analysis. Hence, we have the following privacy guarantee for Algorithm 1.

Lemma 3. *Algorithm 1 satisfies (α, ϵ)-RDP with*

$$\epsilon = \log\left(\sum_{j=1}^{m} \frac{1}{m} \cdot \exp((\alpha-1)\cdot\epsilon_\alpha(\Delta_2^*[j], \Delta_1^*[j], N\lambda))\right),$$

where $\epsilon_\alpha(\Delta_2, \Delta_1, \lambda)$ is defined as in Eq. (3)

Challenge in combining Skellam with RSGD-AR. The main challenge in combining Skellam with RSGD-AR is that Skellam noise assumes "worst-case" analysis due to discretization/rounding while RSGD-AR explores the "average-case" sensitivity due to random permutation. To be more specific, note that the condition used for the conditional rounding step in previous work [40] is dependent on the sensitivity of the original vector, which depends on the randomness of permutation in our case. However, such information can not be learnt by anyone, since no one knows which record is the differing one for the neighboring dataset (recall that the definition of DP holds for any neighboring inputs). In addition, naively setting the condition to be dependent on $\boldsymbol{\Delta}[1]$ even leads to privacy violation, since the differing record may not be in the first batch. Hence, we choose to use a condition that is independent of the random permutation, and is only dependent on the dimension of the data, simplifying the privacy analysis.

Advantage over the existing solution. First, we recall the reason for combining integer-valued noise with SecAgg under FL is to provide central-DP-like privacy guarantees rather than local DP. However, SecAgg also introduces prohibitive communication overhead, which is a major concern in FL, especially when running iterative algorithms (e.g., distributed SGD). To alleviate the communication issue, Sk-RSGD-AR utilizes the smoothness and convexity of the loss function LR, which enables direct computation for the parameter-space sensitivity of SGD, based on the results from [19]. In Sk-RSGD-AR, clients only perturb their locally trained model parameters and then securely aggregate the perturbed models only once, with provable DP guarantees. Compared with Sk-SGD, our solution reduces the number of communication rounds *from thousands to only one* while achieving a comparable privacy-utility trade-off, as we will see in the next section.

Computation costs. At first sight, it seems that Sk-RSGD-AR incurs extra computation costs during the training due to the discretization step. We would like to note that such computation costs only incur once at the end of the FL process. Its competitor Sk-SGD [3] requires such computation at every iteration of the training process. Finally, we remark that the only overhead of Sk-RSGD-AR compared with Sk-SGD is that Sk-RSGD-AR incurs extra memory costs on the clients for storing model parameters from past iterations for averaging, which is affordable since LR models are small.

5 EXPERIMENTS

We evaluate the performance of our proposed solution on four datasets with different scales, *Credit Approval*, *MAGIC Gamma Telescope*, *Adult*, and *ACSIncome* [24, 26, 35], which contain about 700 instances and 15 attributes, 20000 instances and 11 attributes, 50000 instances and 14 attributes, 500000 instances and 700 attributes, respectively. For the ACSIncome dataset, we use the data collected from the five states (corresponding to five clients), California, Texas, New York, Florida, and Michigan in the year 2017. The original data of the five states contain different attributes and we take their intersection. We randomly partition each dataset into training/validation/test splits by $6:2:2$. As for different attributes, we normalize the numeric ones into $[0, 1]$, and pre-process the categorical attributes by one-hot embedding as in [19].

Baselines. We see DPSGD [1] and RSGD-AR [19] as the strong baselines under the centralized setting. The details of the compared algorithms are listed as follows.

- Non-dp baseline: The non-dp baseline (SGD) randomly samples a batch of records from the input dataset and performs gradient updates on the sampled batch. The algorithm is repeated until convergence.
- DPSGD: DPSGD [1] is essentially similar to the non-dp baseline, except that before updating the model parameters, we first perturb the gradient sum with additive Gaussian noise. Accordingly, the model parameters are updated using the perturbed gradient.
- RSGD-AR: The main difference between RSGD-AR [19] and DPSGD [1] is that in RSGD-AR, we only inject random Gaussian noise into the final model parameters. As we have mentioned, RSGD-AR also achieves DP, by exploiting the convexity and smoothness condition of the loss function of LR. Here we regard both DPSGD and RSGD-AR under the centralized setting as the strong baselines.
- Sk-SGD: Sk-SGD can be seen as the distributed version of DPSGD. The difference is that clients inject symmetric Skellam noise into their local gradients and use SecAgg to aggregate the noisy gradients. The server then uses the aggregate noisy gradient to update the model parameters. Note that the original Sk-SGD was proposed in the distributed DP literature, which focuses on client-level DP instead of

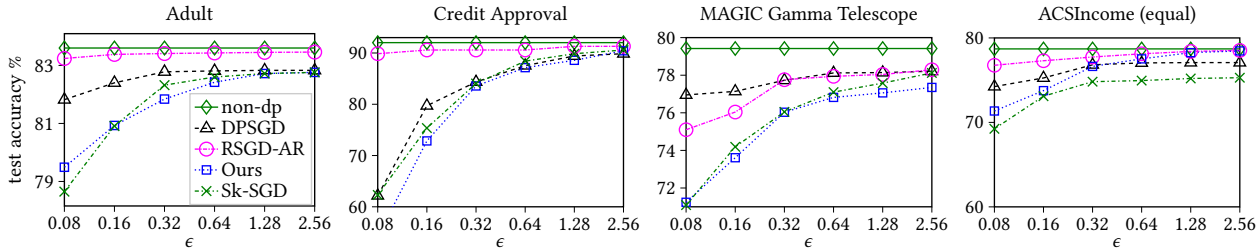

Figure 1: Test accuracy for multiple datasets under varying privacy parameter ϵ, with δ fixed to 10^{-8}.

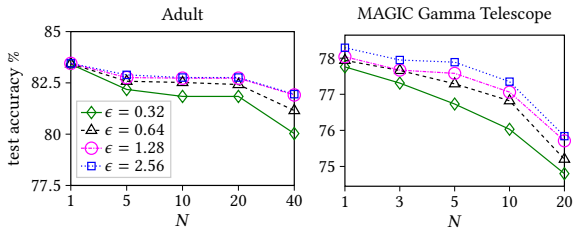

Figure 2: Test accuracy of our solution Sk-RSGD-AR with varying number of clients N.

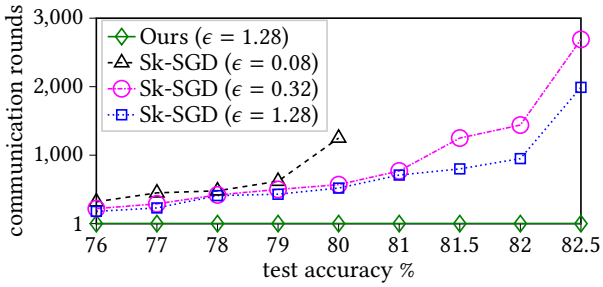

Figure 3: Number of communication rounds to achieve target test accuracy on *Adult* under varying ϵ.

record/item-level DP. Here we adopt their algorithm into our setting, which focuses on record/item-level DP. We see Sk-SGD [3] as our competitor under the distributed setting.

Hyper-parameters. For the relatively small datasets *Credit Approval* and *MAGIC Gamma Telescope*, we assume there are 5 and 10 clients, respectively. For *Adult*, we assume there are 20 clients. Under the distributed setting, the clients partition the training dataset uniformly at random. For *ACSIncome*, we assume there are 5 clients (corresponding to 5 states), as we have mentioned. We would like to mention that we did not tune the hyperparameters in favor of any algorithms reported in this work. For all algorithms, we fix the $\mathcal{L}_2$ clipping norm for an individual gradient to 1 and the weight decay of the loss function to 0.001. For both Sk-SGD and Sk-RSGD-AR, the server's learning rate is fixed at 1. For DPSGD (resp. its distributed version Sk-SGD), we fix the learning rate (resp. local learning rate) to 0.1. For DPSGD, Sk-SGD, and our solution, we fix the subsampling rate to 0.0034 at every SGD iteration. We vary the privacy parameter ϵ from $\{0.08, 0.16, 0.32, 0.64, 1.28\}$ and fix δ to 10^{-8}, and report the average test accuracy for each setting over 10 runs.

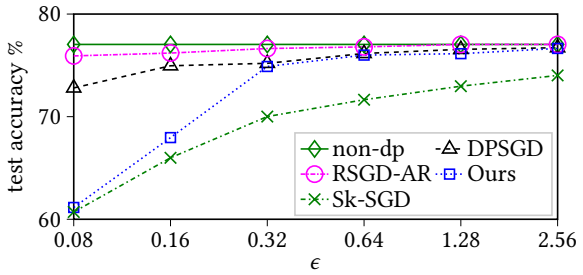

Figure 4: Test accuracy for ACSIncome with non-equal partitions.

Figure 1 shows the comparison between Sk-RSGD-AR with other baselines in different datasets. As stated above, we take the performance of the baselines in centralized setting, non-dp, DPSGD and RSGD-AR as our ground truth. We observe that with increasing privacy budget ϵ, the gap between ground truth and baselines in distributed setting narrows, and Sk-RSGD-AR achieves comparable performance with the the competitor Sk-SGD, but with fewer communication costs. In Figure 1, when the privacy budget ϵ is sufficient, compared with the non-dp, DPSGD, and RSGD-AR in centralized setting Sk-RSGD-AR only incurs 1%-3% drop in test accuracy.

As for communication costs, we also plot the number of communication rounds to achieve a given test accuracy for *Adult* dataset in Figure 3. We can see that Sk-RSGD-AR takes only **one** round of communication to achieve a test accuracy of 82.5% under $\epsilon = 1.28$, whereas for Sk-SGD, the number of communication rounds increases (up to 2690) as the target test-accuracy increases under different privacy parameters. This is because Sk-RSGD-AR is based on output perturbation, and all clients communicate only once at the very end of the training process, as we have mentioned in Section 4. We omit the results on other datasets as Sk-RSGD-AR takes only **one** round of communication, regardless of the input.

Effect of varying N. For *MAGIC Gamma Telescope* and *Adult*, we also test the effectiveness of our solution with varying numbers of clients from 1 to 40 under different settings of ϵ and report the average test accuracy over ten runs. In particular, $N = 1$ corresponds to the centralized setting. The results are shown in Figure 2. When the number of clients is larger than 20 in Adult, the performance declines due to the fact that models trained on small local dataset portions do not converge. While in MAGIC Gamma Telescope, the

test accuracy is much more sensitive, and declines at the very beginning. As for different privacy budgets, Sk-RSGD-AR with larger ϵ is much more robust.

Partition of the dataset. In our experiments, we have assumed that there are an equal number of private data samples for all clients. Such an assumption is commonly used in the experiments of FL [49, 50]. We are aware that such an assumption may not hold in practice. Given those circumstances, we require each client to first scale her model parameters before perturbation and aggregation. The scale ratio for each client is based on her portion of the entire dataset. This pre-processing step is to make sure that each client contributes to the global model with weights proportional to her number of data samples. Note that with such scaling, the level of privacy guarantee for the individual records of different clients may be different. In particular, the clients with the smallest numbers of records preserve the highest level (smallest ϵ) of DP for their records. We report the test accuracy on ACSIncome with non-equal partitions in Figure 4. The reported level of ϵ for our solution is computed for the client with the largest number of records (i.e., California). We note that even under such an unfair setting, Sk-RSGD-AR still outperforms its competitor Sk-SGD by a large margin. We have open-sourced our implementation in https://github.com/DavdGao/FLDP.

6 RELATED WORKS

Differentially private federated learning. We review the differentially private federated learning literature next. [6] propose a random check-in distributed protocol and show that random participation made on the client side improves the privacy guarantee. [50, 58] train large-scale models under the distributed setting with formal DP guarantees. [11] build a system for FL. [39] also study the privacy-preserving Empirical Risk Minimization problem under the distributed setting. Different from our approach, they use continuous additive noise to enforce DP. Note that continuous noise is not directly compatible with MPC or SecAgg, which operate on finite integer domains. Hence, the methods from [39] are not directly comparable with our method. Differentially private FL with one-shot communication is also studied in [45, 57, 67] with the assumption of access to an unlabeled public dataset. In contrast, our target problem is fundamentally different from theirs. In our setting, we consider every training data sample used to obtain the final model as sensitive information and hence, provide a DP guarantee for it. Indeed, publicly accessible data could improve the performance of a DP model. But we would also like to point out that having accessible unlabeled public datasets might not be the appropriate assumption in our problem setting. In application fields such as governmental collaboration, health care, and finance, even unlabeled data is hardly accessible to the public as the unlabeled data still contains a significant amount of sensitive information for an individual. We refer readers to [56] for a survey in FL with DP.

Output perturbation v.s. gradient perturbation. Two main approaches used for logistic regression (and also general ML applications) are output perturbation [19] and gradient perturbation [1, 62]. The idea of gradient perturbation is to inject noise into the gradients of the gradient descent algorithm, to guarantee that the release of the noisy gradients is DP in every iteration. The overall privacy guarantee follows from the composition theorem for differential privacy. Output perturbation, on the other hand, injects DP noise directly into the final output of an algorithm. The focus of this paper is on logistic regression, which has a smooth and convex loss function that eases the sensitivity analysis. Accordingly, in the centralized setting, it has been shown that output perturbation leads to better privacy-utility trade-offs than gradient perturbation [19]. However, unlike gradient perturbation, output perturbation has fewer applications as the sensitivity of the output of an algorithm can not always be computed easily.

Besides the aforementioned gradient perturbation [1, 62] and output perturbation [19], objective perturbation [18] directly injects real-valued noise to the objective in Eq. (1). The trusted server then solves the perturbed optimization problem. They show that the optimal solution to the perturbed objective satisfies DP. However, it is difficult to adapt objective perturbation to distributed DP settings, since under distributed DP no one can be trusted with the perturbed objective function, which still contains sensitive information of the private dataset. We also note that the recent work [21] provides tighter analyses for gradient perturbation techniques under the centralized setting. Similar to objective perturbation, however, their analysis is not directly applicable to the distributed DP setting. Adapting both objective perturbation and the advanced analysis for DPSGD to the distributed setting is a promising future work direction. We are also aware there are other techniques in the privacy and learning literature to reduce communication costs, such as [2, 13, 20, 43, 60]. Our mechanism incurs low communication costs in the sense that it only requires one round of communication. Further combining these techniques with our solution is a promising future work direction.

7 CONCLUSION

In this work, we propose Sk-RSGD-AR, a communication efficient and differentially private solution for logistic regression under the distributed setting. Sk-RSGD-AR extends the original output perturbation technique [19] from the centralized setting to the distributed setting. Compared with the baseline solution, Sk-RSGD-AR achieves comparable privacy-utility trade-offs (if not better) on multiple real-world datasets under a variety of parameter settings. In addition, Sk-RSGD-AR incurs a much lower communication cost than the existing solution, which we believe is of great importance in the applications of FL. Regarding future work, we plan to further reduce the communication cost of differentially private federated learning by incorporating dimensionality reduction techniques. In addition, practical threat models that include malicious participants in FL, and FL applications for vertically partitioned databases are also promising research directions that are worth looking into.

ACKNOWLEDGMENTS

This research is supported by the National Research Foundation, Singapore under its Industry Alignment Fund - Pre-positioning (IAF-PP) Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore. We would like to thank the anonymous reviewers and Hongyan Chang for their constructive feedback.

REFERENCES

[1] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *CCS*. 308–318.

[2] Reza Abbasi Asl and Bin Yu. 2021. Structural Compression of Convolutional Neural Networks with Applications in Interpretability. *Frontiers in Big Data* 4 (08 2021).

[3] Naman Agarwal, Peter Kairouz, and Ziyu Liu. 2021. The Skellam Mechanism for Differentially Private Federated Learning. In *NeurIPS 2021*. 5052–5064.

[4] Naman Agarwal, Ananda Theertha Suresh, Felix Yu, Sanjiv Kumar, and H. Brendan McMahan. 2018. CpSGD: Communication-Efficient and Differentially-Private Distributed SGD. In *NeurIPS*. 7575–7586.

[5] Apple. 2016. *Differential Privacy Overview*. Retrieved December 21, 2020 from https://www.apple.com/privacy/docs/Differential_Privacy_Overview.pdf

[6] Borja Balle, Peter Kairouz, H. Brendan McMahan, Om Thakkar, and Abhradeep Thakurta. 2020. Privacy Amplification via Random Check-Ins. In *NeurIPS*.

[7] Borja Balle and Yu-Xiang Wang. 2018. Improving the Gaussian Mechanism for Differential Privacy: Analytical Calibration and Optimal Denoising. In *ICML*, Vol. 80. 403–412.

[8] Raef Bassily and Adam Smith. 2015. Local, Private, Efficient Protocols for Succinct Histograms. In *STOC*. 127–135.

[9] Raef Bassily, Adam D. Smith, and Abhradeep Thakurta. 2014. Private Empirical Risk Minimization: Efficient Algorithms and Tight Error Bounds. *FOCS* (2014), 464–473.

[10] James Henry Bell, Kallista A. Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. 2020. Secure Single-Server Aggregation with (Poly)Logarithmic Overhead. In *CCS*. 1253–1269.

[11] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. 2019. Towards Federated Learning at Scale: System Design. In *MLSys*, Vol. 1. 374–388.

[12] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *CCS*. 1175–1191.

[13] Keith Bonawitz, Fariborz Salehi, Jakub Konečný, Brendan McMahan, and Marco Gruteser. 2019. Federated Learning with Autotuned Communication-Efficient Secure Aggregation. In *ACSSC*. 1222–1226.

[14] Léon Bottou, Frank E Curtis, and Jorge Nocedal. 2018. Optimization methods for large-scale machine learning. *Siam Review* 60, 2 (2018), 223–311.

[15] Clément L. Canonne, Gautam Kamath, and Thomas Steinke. 2020. The Discrete Gaussian for Differential Privacy. In *NeurIPS*.

[16] Hongyan Chang, Virat Shejwalkar, Reza Shokri, and Amir Houmansadr. 2019. Cronus: Robust and Heterogeneous Collaborative Learning with Black-Box Knowledge Transfer. arXiv:1912.11279 [stat.ML]

[17] Hongyan Chang and Reza Shokri. 2023. Bias Propagation in Federated Learning. In *ICLR*.

[18] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. 2011. Differentially Private Empirical Risk Minimization. *Journal of machine learning research : JMLR* 12 (2011), 1069–1109.

[19] Chen Chen, Jaewoo Lee, and Dan Kifer. 2019. Renyi Differentially Private ERM for Smooth Objectives. In *AISTATS*. 2037–2046.

[20] Wei-Ning Chen, Christopher A. Choquette-Choo, Peter Kairouz, and Ananda Theertha Suresh. 2022. The Fundamental Price of Secure Aggregation in Differentially Private Federated Learning. In *ICML*.

[21] Rishav Chourasia, Jiayuan Ye, and Reza Shokri. 2021. Differential privacy dynamics of langevin diffusion and noisy gradient descent. *NeurIPS* 34 (2021), 14771–14781.

[22] David R Cox. 1958. The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)* 20, 2 (1958), 215–232.

[23] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting Telemetry Data Privately. In *NeurIPS*. 3574–3583.

[24] Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. 2021. Retiring Adult: New Datasets for Fair Machine Learning. In *NeurIPS*. 6478–6490.

[25] Irit Dinur and Kobbi Nissim. 2003. Revealing Information While Preserving Privacy. In *PODS*. 202–210.

[26] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[27] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *TCC*. 265–284.

[28] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* 9, 3–4 (aug 2014), 211–407.

[29] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. 2010. Boosting and differential privacy. In *FOCS*. 51–60.

[30] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. 2019. Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity. In *SODA*. 2468–2479.

[31] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *CCS*. 1054–1067.

[32] V. Feldman, I. Mironov, K. Talwar, and A. Thakurta. 2018. Privacy Amplification by Iteration. In *FOCS*. 521–532.

[33] Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. 2010. Differentially Private Combinatorial Optimization. In *SODA*. 1106–1125.

[34] Moritz Hardt and Guy N. Rothblum. 2010. A Multiplicative Weights Mechanism for Privacy-Preserving Data Analysis. In *FOCS*. 61–70.

[35] D. Heck, Johannes Knapp, J-N. Capdevielle, George C. Schatz, and T. J. Thouw. 1998. CORSIKA: A Monte Carlo code to simulate extensive air showers.

[36] A. Hedayat and W. D. Wallis. 1978. Hadamard Matrices and Their Applications. *The Annals of Statistics* 6, 6 (1978), 1184 – 1238.

[37] Briland Hitaj, Giuseppe Ateniese, and Fernando Pérez-Cruz. 2017. Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning. In *CCS*. 603–618.

[38] Rui Hu, Yuanxiong Guo, and Yanmin Gong. 2021. Concentrated Differentially Private Federated Learning With Performance Analysis. *IEEE Open Journal of the Computer Society* 2 (2021), 276–289.

[39] Bargav Jayaraman, Lingxiao Wang, David Evans, and Quanquan Gu. 2018. Distributed Learning without Distress: Privacy-Preserving Empirical Risk Minimization. In *NeurIPS*.

[40] Peter Kairouz, Ziyu Liu, and Thomas Steinke. 2021. The Distributed Discrete Gaussian Mechanism for Federated Learning with Secure Aggregation. In *ICML*. 5201–5212.

[41] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista A. Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaïd Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. 2021. Advances and Open Problems in Federated Learning. *Found. Trends Mach. Learn.* 14, 1-2 (2021), 1–210.

[42] J. Kiefer and Jacob Wolfowitz. 1952. Stochastic Estimation of the Maximum of a Regression Function. *Annals of Mathematical Statistics* 23 (1952), 462–466.

[43] Heiner Kirchhoffer, Paul Haase, Wojciech Samek, Karsten Müller, Hamed Rezazadegan-Tavakoli, Francesco Cricri, Emre B. Aksu, Miska M. Hannuksela, Wei Jiang, Wei Wang, Shan Liu, Swayambhoo Jain, Shahab Hamidi-Rad, Fabien Racapé, and Werner Bailer. 2022. Overview of the Neural Network Compression and Representation (NNR) Standard. *IEEE Transactions on Circuits and Systems for Video Technology* 32, 5 (2022), 3203–3216.

[44] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated Learning: Strategies for Improving Communication Efficiency. https://arxiv.org/abs/1610.05492

[45] Qinbin Li, Bingsheng He, and Dawn Xiaodong Song. 2020. Practical One-Shot Federated Learning for Cross-Silo Setting. In *International Joint Conference on Artificial Intelligence*.

[46] P. McCullagh and J.A. Nelder. 1989. *Generalized Linear Models, Second Edition*. Chapman & Hall. http://books.google.com/books?id=h9kFH2_FfBkC

[47] Ryan McKenna and Daniel Sheldon. 2020. Permute-and-Flip: A New Mechanism for Differentially Private Selection. In *NeurIPS*. 11 pages.

[48] Ryan Mckenna, Daniel Sheldon, and Gerome Miklau. 2019. Graphical-model based estimation and inference for differential privacy. In *ICML*. 4435–4444.

[49] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*. 1273–1282.

[50] Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning Differentially Private Recurrent Language Models. In *ICLR*.

[51] H. B. McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*.

[52] Frank McSherry and Kunal Talwar. 2007. Mechanism Design via Differential Privacy. In *FOCS*. 94–103.

[53] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting Unintended Feature Leakage in Collaborative Learning. In *S&P*. 691–706.

[54] Ilya Mironov. 2017. Rényi Differential Privacy. In *CSF*. 263–275.

[55] Joe Near. 2018. Differential Privacy at Scale: Uber and Berkeley Collaboration. In *Enigma*.

[56] Ahmed El Ouadrhiri and Ahmed Abdelhadi. 2022. Differential Privacy for Deep and Federated Learning: A Survey. *IEEE Access* 10 (2022), 22359–22380.

[57] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. 2018. Scalable Private Learning with PATE. In *ICLR*.
[58] Swaroop Indra Ramaswamy, Om Thakkar, Rajiv Mathews, Galen Andrew, H. B. McMahan, and Franccoise Beaufays. 2020. Training Production Language Models without Memorizing User Data. *ArXiv* abs/2009.10031 (2020).
[59] Herbert Robbins and Sutton Monro. 1951. A Stochastic Approximation Method. *The Annals of Mathematical Statistics* 22, 3 (1951), 400–407.
[60] Abhin Shah, Wei-Ning Chen, Johannes Ballé, Peter Kairouz, and Lucas Theis. 2022. Optimal Compression of Locally Differentially Private Mechanisms. In *ICAIS*. 7680–7723.
[61] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-Preserving Deep Learning. In *CCS*. 1310–1321.
[62] Shuang Song, Kamalika Chaudhuri, and Anand D. Sarwate. 2013. Stochastic gradient descent with differentially private updates. In *Global SIP*. 245–248.
[63] Tim van Erven and Peter Harremoës. 2014. Rényi Divergence and Kullback-Leibler Divergence. *IEEE Trans. Inf. Theory* 60, 7 (2014), 3797–3820.
[64] Di Wang, Minwei Ye, and Jinhui Xu. 2017. Differentially Private Empirical Risk Minimization Revisited: Faster and More General. In *NeurIPS*, Vol. 30.
[65] David P. Woodruff. 2014. Sketching as a Tool for Numerical Linear Algebra. *Found. Trends Theor. Comput. Sci.* 10, 1-2 (2014), 1–157. https://doi.org/10.1561/0400000060
[66] Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2017. PrivBayes: Private Data Release via Bayesian Networks. *TODS* 42, 4 (2017).
[67] Yuqing Zhu, Xiang Yu, Yi-Hsuan Tsai, Francesco Pittaluga, Masoud Faraki, Manmohan chandraker, and Yu-Xiang Wang. 2022. Voting-based Approaches For Differentially Private Federated Learning. In *NeurIPS 2022 Workshop Federated Learning*.