Achieving Differential Privacy in Vertically Partitioned Multiparty Learning

Depeng Xu
University of Arkansas
Fayetteville, AR, USA
depengxu@uark.edu

Shuhan Yuan
Utah State University
Logan, UT, USA
Shuhan.Yuan@usu.edu

Xintao Wu
University of Arkansas
Fayetteville, AR, USA
xintaowu@uark.edu

2021 IEEE International Conference on Big Data (Big Data) | 978-1-6654-3902-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/BigData52589.2021.9671502

***Abstract*—Preserving differential privacy has been well studied under the centralized setting. However, it's very challenging to preserve differential privacy under multiparty setting, especially for the vertically partitioned case. In this work, we propose a new framework for differential privacy preserving multiparty learning in the vertically partitioned setting. Our core idea is based on the functional mechanism that achieves differential privacy of the released model by adding noise to the objective function. We show the server can simply dissect the objective function into single-party and cross-party sub-functions, and allocate computation and perturbation of their polynomial coefficients t o l ocal p arties. O ur m ethod n eeds o nly o ne r ound of noise addition and secure aggregation. The released model in our framework achieves the same utility as applying the functional mechanism in the centralized setting. Evaluation on real-world and synthetic datasets for linear and logistic regressions shows the effectiveness of our proposed method.**

***Index Terms*—differential privacy, federated learning, vertical partition**

I. Introduction

Rapid growth of modern technology is largely driven by data. In most industries, data exist in the form of isolated islands. Federated learning is proposed to build machine learning models based on distributed datasets across multiple parties [1], [2]. In particular, vertically partitioned multiparty learning is applicable when parties share the same record ID space but differ in the feature space, such as using user experience on the web to support decisions on healthcare. Understandably, parties do not want to share raw data or statistics due to privacy concerns. How to build an efficient global model through data barrier while preserving local parties' privacy is a challenging problem.

Differential privacy is a standard privacy preserving scheme to achieve opt-out right of individuals [3]. In general, differential privacy guarantees the query results or the released model cannot be exploited by attackers to derive whether one particular record is present or absent in the underlying dataset. Many mechanisms have been proposed to achieve differential privacy [4]–[7]. For example, the classic Laplace mechanism injects random noise into the released results such that the inclusion or exclusion of a single record makes no statistical difference [3]. For machine learning models, research in [8], [9] develops methods of adding noise to gradients to preserve differential privacy of training data. Functional mechanism [10], which adds noise to the objective function rather than parameters of built models, has also been shown great success in deep learning [11].

Recently, several works propose to train privacy preserving models in the decentralized settings. Research in [12] proposes a collaborative deep learning framework in which participants train independently and share only subsets of updates of parameters in the horizontally distributed setting. However, it is not applicable in the vertically partitioned setting. This is because we cannot partition the gradients based on features and each local party needs to collect raw data of those features owned by other parties, which requires extensive use of secure multiparty computation to update gradients in each iteration.

There have been several research works on building privacy preserving models in the vertically partitioned setting. Research in [13] develops a framework for private data sharing for the purpose of statistical estimation. Each party communicates perturbed random projections of their locally held features to ensure differential privacy. However, the task focuses on the statistical estimation of coefficients rather than releasing a jointly trained model in our context. Research in [14] develops a distributed private block-coordinate Frank-Wolfe algorithm under arbitrary sampling. They design an active feature sharing scheme by utilizing private Johnson-Lindenstrauss transform to update local partial gradients in a differentially private and communication efficient manner. However, the gradient perturbation requires noise addition in each iteration, which is difficult to achieve good utility-privacy tradeoff, as shown in our evaluation. In ensemble learning, research in [15] proposes to enhance privacy preserving logistic regression by feature-wise partitioned stacking. The proposed method is combined with hypothesis transfer learning to enable learning across different organizations. However, this research does not really apply to vertical partitioned learning as the high-level model still needs to access all data to construct meta-data set when training private logistic regression.

In this work, we propose a new framework for differential privacy preserving multiparty learning in the vertically partitioned setting. Our core idea is based on the functional mechanism that achieves differential privacy of the released model by adding noise to the objective function. In the framework, we show the server can simply dissect the objective function into

single-party and cross-party sub-functions and rewrite them in the polynomial form. For the coefficients in the polynomial form related to one single party, they can be calculated by each party in a differentially private manner. For those coefficients related to two or multiple parties, we apply secure vector multiplication and then add noise before sending to server. The server then solves the perturbed objective function in the server side and releases the private model. Our method needs only one round of noise addition and secure aggregation. Hence, both good privacy-utility tradeoff and computational efficiency can be achieved. In fact, the released model in our framework achieves the same utility as applying the functional mechanism in the centralized setting. Our evaluation on real-world and synthetic datasets for linear and logistic regressions shows the effectiveness of our proposed method. Our novelty and contributions are summarized below:

(1) We propose a novel framework to achieve differential privacy in the vertically partitioned setting. The developed method perturbs the single-party and cross-party coefficients in one single round and hence achieves differential privacy for all parties with minimal communication cost.
(2) Our approach adds less noise than state-of-the-art approaches to achieve the same level of differential privacy, e.g., we reduce noise addition by a magnitude of the number of total iterations compared with gradient perturbation approaches.
(3) Our framework only needs one round of secure vector multiplication to prevent the disclosure of raw data between the server and the participating parties. All secure vector multiplications occur at the beginning of our approach. We use the standard MPC protocol based on Shamir's secret sharing scheme and evaluate its execution performance in our framework.
(4) Different from the horizontally partitioned setting, the levels of differential privacy achieved with respect to global data and local data are different. Our work is the first to study this problem. We provide theoretical analysis on the levels of differential privacy achieved by our algorithm with respect to global data (Theorem 1) and local data (Theorem 2).
(5) We propose two protocols, top-down protocol when the server selects the privacy budget (Section III-A to III-C), and bottom-up protocol when local parties select their privacy budgets (Section III-D).

The rest of this paper is organized as follows. Section II introduces backgrounds on differential privacy, functional mechanism, and secure vector multiplication. Section III shows our framework of achieving differential privacy in the vertically partitioned multiparty learning based on functional mechanism. Section IV reports our evaluation results in terms of privacy-utility tradeoff and execution time. Finally, the conclusions are summarized in Section V.

II. Preliminaries

In this section, we revisit how to achieve differential privacy in the centralized setting. Consider a dataset D with n users. Each user's information is a record $t_i = \{\mathbf{x}_i, y_i\}$, where $\mathbf{x}_i$ is the user's feature information and y_i is the user's label. The total number of features is d. We assume that $x_{ia} \in [-1, 1]$ for $a \in [1, d]$ and $y \in [-1, 1]$ for linear regression or $y \in \{0, 1\}$ for logistic regression. The objective is to build a model $\hat{y} = q(\mathbf{x}; \mathbf{w})$ from dataset D that achieves differential privacy. To fit the model parameter $\mathbf{w}$, we have an objective function $f_D(\mathbf{w}) = \sum_{i=1}^{n} f(t_i; \mathbf{w})$ that takes t_i and $\mathbf{w}$ as input. The optimal model parameter $\mathbf{w}$ is defined as: $\mathbf{w} = \arg\min_{\mathbf{w}} \sum_{i=1}^{n} f(t_i; \mathbf{w})$. We use linear regression and logistic regression as examples in this paper.

A. Differential Privacy

Differential privacy guarantees output of a query q be insensitive to the presence or absence of one record in a dataset.

Definition 1. *Differential privacy [3]. A mechanism $\mathcal{M}$ satisfies ε-differential privacy, if for all neighboring datasets D and D' that differ in exactly one record and all subsets Z of $\mathcal{M}$'s range:*

$$\Pr(\mathcal{M}(D) \in Z) \leq \exp(\varepsilon) \cdot \Pr(\mathcal{M}(D') \in Z).$$

The parameter ε denotes the privacy budget (smaller values indicate stronger privacy guarantee).

Definition 2. *Global sensitivity [3]. Given a query q: $D \rightarrow \mathbb{R}^d$, the global sensitivity Δ is defined as $\Delta = \max_{D,D'} ||q(D) - q(D')||_1$.*

The global sensitivity measures the maximum possible change in $q(D)$ when one record in the dataset changes. The Laplace mechanism is a popular method to achieve differential privacy. It adds identical independent noise into each output value of $q(D)$.

Definition 3. *Laplace mechanism [3]. Given a dataset D and a query q, a mechanism $\mathcal{M}(D) = q(D) + \boldsymbol{\eta}$ satisfies ε-differential privacy, where $\boldsymbol{\eta}$ is a random vector drawn from $Lap(\frac{\Delta}{\varepsilon})$ [1].*

Alternately, adding Gaussian noise $N(0, \sigma^2)$ with σ calibrated to $\Delta \ln(1/\delta)/\varepsilon$, one can achieve (ε, δ)-differential privacy, where $\delta > 0$ gives relaxed differential privacy.

B. Functional Mechanism

Functional mechanism [10] is a differentially private method designed for optimization-based models. It achieves ε-differential privacy by injecting noise into the objective function and returns privacy preserving parameter $\bar{\mathbf{w}}$ that minimizes the perturbed objective function.

Because the objective function $f_D(\mathbf{w})$ is a complicated function of $\mathbf{w}$, the functional mechanism exploits the polynomial representation of $f_D(\mathbf{w})$. The model parameter $\mathbf{w}$ is a vector that contains d values $w_1, w_2, \cdots, w_d$. Let $\phi(\mathbf{w})$ denote

[1]The Laplace distribution $Lap(\boldsymbol{\eta}|\mu, \sigma)$ with mean μ and scale σ has probability density function $Lap(\eta|\mu, \sigma) = \frac{1}{2\sigma} \exp(\frac{|x-\mu|}{\sigma})$. Its variance is $2\sigma^2$. Note $\mu = 0$ if not specified.

a product of $w_1, w_2, \cdots, w_d$, i.e., $\phi(\mathbf{w}) = w_1^{c_1} \cdot w_2^{c_2} \cdots w_d^{c_d}$ for some $c_1, c_2, \cdots, c_d \in \mathbb{N}$. Let $\mathbf{\Phi}_j (j \in \mathbb{N})$ denote the set of all products of $w_1, w_2, \cdots, w_d$ with degree j, i.e., $\mathbf{\Phi}_j = \{w_1^{c_1} w_2^{c_2} \cdots w_d^{c_d} | \sum_{l=1}^{d} c_l = j\}$. For example, $\mathbf{\Phi}_1 = \{w_1, w_2, \cdots, w_d\}$, and $\mathbf{\Phi}_2 = \{w_a \cdot w_b | a, b \in [1, d]\}$.

Based on the Stone-Weierstrass Theorem [16], any continuous and differentiable function can be expressed in the polynomial representation. Hence, the objective function $f_D(\mathbf{w})$ can be expressed as a polynomial of $w_1, w_2, \cdots, w_d$, for some $J \in \mathbb{N}$:

$$f_D(\mathbf{w}) = \sum_{i=1}^{n} \sum_{j=0}^{J} \sum_{\phi \in \mathbf{\Phi}_j} \lambda_{\phi t_i} \phi(\mathbf{w}), \tag{1}$$

where $\lambda_{\phi t_i} \in \mathbb{R}$ denotes the coefficient of $\phi(\mathbf{w})$.

Functional mechanism perturbs the objective function $f_D(\mathbf{w})$ by injecting Laplace noise into its polynomial coefficients $\bar{\lambda}_\phi = \sum_{i=1}^{n} \lambda_{\phi t_i} + Lap(\frac{\Delta_f}{\varepsilon})$, where the global sensitivity of $f_D(\mathbf{w})$ is

$$\Delta_f = 2 \max_t \sum_{j=1}^{J} \sum_{\phi \in \mathbf{\Phi}_j} ||\lambda_{\phi t}||_1.$$

Then the model parameter $\bar{\mathbf{w}}$ is derived by minimizing the perturbed function $\bar{f}_D(\mathbf{w})$.

1) Application to Linear Regression: A linear regression on D returns a prediction function $\hat{y}_i = q(\mathbf{x}_i; \mathbf{w}) = \mathbf{x}_i^T \mathbf{w}$. The objective function of linear regression is defined as:

$$\begin{aligned} f_D(\mathbf{w}) = & \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \mathbf{w})^2 \\ = & \sum_{i=1}^{n} (y_i)^2 - \sum_{a=1}^{d} (2 \sum_{i=1}^{n} y_i x_{ia}) w_a \\ & + \sum_{1 \le a,b \le d} (\sum_{i=1}^{n} x_{ia} x_{ib}) w_a \cdot w_b. \end{aligned} \tag{2}$$

We get different order polynomial coefficients $\lambda_{\phi_0} = \sum_{i=1}^{n} (y_i)^2$, $\lambda_{w_a} = -2 \sum_{i=1}^{n} y_i x_{ia}$, and $\lambda_{w_a \cdot w_b} = \sum_{1 \le a,b \le d} \sum_{i=1}^{n} x_{ia} x_{ib}$. Then we add $Lap(\frac{\Delta_f}{\varepsilon})$ to the coefficients, where the global sensitivity of $f_D(\mathbf{w})$ for linear regression is

$$\Delta_f = 2(1 + 2d + d^2).$$

2) Application to Logistic Regression: A logistic regression on D returns a function which predicts $\hat{y}_i = 1$ with probability $\hat{y}_i = q(\mathbf{x}_i; \mathbf{w}) = \exp(\mathbf{x}_i^T \mathbf{w}) / (1 + \exp(\mathbf{x}_i^T \mathbf{w}))$. The objective function of logistic regression is defined as:

$$f_D(\mathbf{w}) = \sum_{i=1}^{n} \left[\log(1 + \exp(\mathbf{x}_i^T \mathbf{w})) - y_i \mathbf{x}_i^T \mathbf{w} \right]. \tag{3}$$

As the polynomial form of Equation 3 contains terms with unbounded degrees, to apply the functional mechanism, it is rewritten as the approximate polynomial representation based on Taylor expansion [10]:

$$f_D(\mathbf{w}) = \Big(\sum_{i=1}^{n} \sum_{j=0}^{2} \frac{f_1^{(j)}(0)}{j!} \left(\mathbf{x}_i^T \mathbf{w} \right)^j \Big) - \Big(\sum_{i=1}^{n} y_i \mathbf{x}_i^T \Big) \mathbf{w}, \tag{4}$$

where $f_1(\cdot) = \log(1 + \exp(\cdot))$, $J = 2$. We get different order polynomial coefficients $\lambda_{w_a} = \sum_{i=1}^{n} \big(\frac{f_1^{(1)}(0)}{1!} - y_i \big) x_{ia}$ and $\lambda_{w_a \cdot w_b} = \sum_{1 \le a,b \le d} \sum_{i=1}^{n} \frac{f_1^{(2)}(0)}{2!} x_{ia} x_{ib}$, and then add $Lap(\frac{\Delta_f}{\varepsilon})$ to the coefficients, where the global sensitivity of $f_D(\mathbf{w})$ for logistic regression is

$$\Delta_f = \frac{d^2}{4} + d.$$

C. Secure Vector Multiplication

Secure multiparty computation (MPC) allows a group of mutually distrustful parties to jointly compute a function over distributed data and provide confidentiality of inputs and intermediate results as well as integrity of the final output. Refer to [17] for a complete introduction. General purpose MPC protocols can run arbitrary computation but impose significant overhead. In our framework, we only need one round of secure vector multiplication and hence we choose one efficient MPC protocol based on Shamir's secret sharing.

Shamir's secret sharing protocols [18] allow a dealer to break a secret value into shares and distribute these shares to group of recipients with the property that any unqualified set of recipients learns nothing about the secret, while any qualified set of recipients can reconstruct the secret from their share. MPC protocols based on Shamir's secret sharing enables two parties to compute the scalar product $\sum_{i=1}^{n} v_i^a \cdot v_i^b$ given the ciphertexts of two vectors $\mathbf{v}^a = \{v_1^a, v_2^a, \ldots, v_n^a\}$ and $\mathbf{v}^b = \{v_1^b, v_2^b, \ldots, v_n^b\}$. Each participant first encrypts its private data and then uploads the ciphertexts to the server. The server then executes the operations over the ciphertexts and returns the encrypted results to the participants. Each pair of participants jointly decrypts the actual result. During this process, the server learns no private data of any participants even if they collude with all the rest participants. Through offloading the computation tasks to the resource-abundant cloud server, this scheme makes the computation and communication complexity on each participant independent to the number of participants.

III. Achieving Differential Privacy in the Vertically Partitioned Multiparty Learning

In this section, we propose a framework of achieving differential privacy in the vertically partitioned multiparty learning based on functional mechanism.

A. Problem Statement

In the vertically partitioned multiparty setting, each user's information is held by K parties separately. Each party P_k owns a disjoint dataset D^k on feature set $\mathbf{X}^k$, where $|\mathbf{X}^k| =$

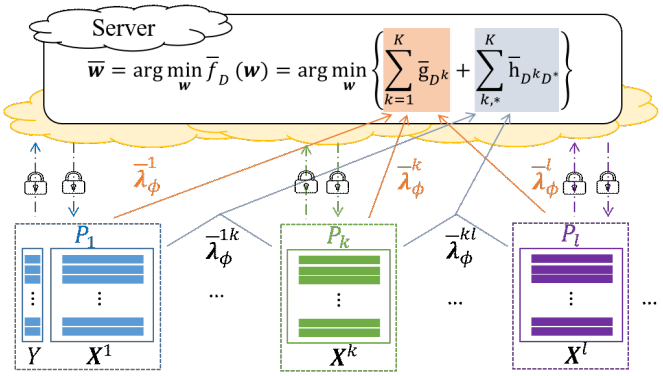

Fig. 1: The framework of achieving differential privacy in the vertically partitioned multiparty learning based on functional mechanism. (1) Dissect the objective function f into sum of the single-party sub-function g and cross-party sub-function h; (2) Collect the private single-party coefficients $\bar{\boldsymbol{\lambda}}_\phi^k$ from each party P_k; (3) Secure vector multiplication and collect the private cross-party coefficients $\bar{\boldsymbol{\lambda}}_\phi^{k*}$; (4) Solve the private objective function $\bar{f}$.

d^k. Similarly, $\mathbf{w}^k$ denotes a subset of $\mathbf{w}$ corresponding to $\mathbf{X}^k$. Label Y is not shared by all parties. Without loss of generality, we simply assume party P_1 holds the label.

A server coordinates K parties to build a multiparty learning model. The server is honest but curious. It aims to release a model trained from the whole dataset D and to ensure the released model satisfies ε-differential privacy w.r.t. D. The parties provide necessary information to the server and help server to build the ε-differentially private global model. But they do not trust the server or each other in terms of sharing users' private information from their local datasets. Each party can share statistics in a differentially private manner. If a computation involves at least two parties, it is conducted by a secure multiparty computation. For party P_1, it shares the label with other parties upon request through secure multiparty computation. On top of that, each party P_k cares about the level of differential privacy achieved w.r.t. its sub-dataset D^k. In the training process, the local party P_k achieves $\varepsilon^{(k)}$-differential privacy, where $\varepsilon^{(k)}$ is ideally a smaller privacy level than ε.

The goal is to reduce the amount of secure multiparty computation and noise addition to the minimum while keeping local information secure and private.

B. General Model Framework

We apply functional mechanism in the vertically partitioned multiparty learning. Functional mechanism does not inject noise directly into the regression results, but ensures privacy by perturbing objective function of the regression analysis. The server only collects information of the objective function at the beginning. The objective function can be dissected based on features, so computation and perturbation of the coefficients can be allocated to local parties by local feature sets. For some operations involving multiple parties, the server conducts secure multiparty computation with the parties. Once the perturbed coefficients are collected from local parties, the server solves the perturbed objective function and releases the private model. Figure 1 illustrates our proposed framework of functional mechanism in the vertically partitioned multiparty learning. The procedure of our framework is shown as Algorithm 1. Overall, there are four steps:

(1) The server dissects objective function f into the sum of single-party sub-function g and cross-party sub-function h, sets and allocates the corresponding coefficients $\{\boldsymbol{\lambda}_\phi^k\}^K$, $\{\boldsymbol{\lambda}_\phi^{k*}\}^K$ for each party P_k to compute (Line 1-2). We use $\boldsymbol{\lambda}_\phi^k, \boldsymbol{\lambda}_\phi^{k*}$ to denote all the single-party coefficients from party P_k and all the cross-party coefficients involving party P_k, respectively. The server calculates the scale of noise needed to achieve ε-differential privacy and informs all the parties (Line 3).

(2) Each party P_k computes polynomial coefficients $\boldsymbol{\lambda}_\phi^k$ in single-party sub-function g from D^k and sends noisy single-party coefficients $\bar{\boldsymbol{\lambda}}_\phi^k$ to the server (Line 6-8).

(3) For polynomial coefficients $\boldsymbol{\lambda}_\phi^{k*}$ in cross-party sub-function h, it involves data from P_k and P_*, such as $\sum_{i=1}^{n} y_i x_{ia}^k$, where y_i is from party P_1, and $\sum_{1\le a,b\le d} \sum_{i=1}^{n} x_{ia}^k x_{ib}^l$, where $X_a \in \mathbf{X}^k, X_b \in \mathbf{X}^l$. Note that $\bar{\lambda}_\phi^{k*}$ is a scalar product of two vectors from party P_k and P_*. All parties send encrypted vectors of user information to the server and receive back the securely aggregated polynomial coefficients $\boldsymbol{\lambda}_\phi^{k*}$ following the MPC protocol (Line 11). The parties add Laplace noise to the results and send $\bar{\boldsymbol{\lambda}}_\phi^{k*}$ back to server (Line 12-13).

(4) The server receives all $\bar{\lambda}_\phi$, solves the noisy objective function $\bar{f}$ and releases the differentially private model (Line 16-18).

1) Dissecting the Objective Function: In our framework, we only need one round of noise addition. The for-loop in Algorithm 1 (Lines 4-15) shows the noise addition and calculation of different subpart/terms of the objective function, all of which together accounts for one single round. In fact, we take advantage that the overall objective function can be dissected into two parts based on features,

$$f_D(\mathbf{w}) = \sum_{k=1}^{K} g_{D^k} + \sum_{1\le k,*\le K} h_{D^k D^*}, \tag{5}$$

where g_{D^k} is the single-party sub-function, and $h_{D^k D^*}$ is the cross-party sub-function. Single-party sub-function g only involves data in party P_k. Cross-party sub-function h involves data in party P_k and at least one other party.

Similarly to objective function $f_D(\mathbf{w})$ in Equation 1, sub-function g and h can also be expressed as polynomials of $w_1, w_2, \cdots, w_d$. For the single-party sub-function,

$$g_{D^k} = \sum_{j=0}^{J} \sum_{\phi \in \boldsymbol{\Phi}_j^k} \lambda_\phi^k \phi(\mathbf{w}^k).$$

Algorithm 1 Functional mechanism in vertically partitioned multiparty learning (D, f, ε)

Set $f_D(\mathbf{w})$ by Equation 5. ▷ Server
Allocate $\{\boldsymbol{\lambda}^k\}^K, \{\boldsymbol{\lambda}^{k*}\}^K$ to parties ▷ Server
Set $\Delta_f = 2\max_t \sum_{j=1}^{J} \sum_{\phi \in \mathbf{\Phi}_j} ||\lambda_{\phi t}||_1$ ▷ Server
for each party P_k **do**
  for each $\lambda_\phi^k \in \boldsymbol{\lambda}_\phi^k$ **do**
    Compute $\lambda_\phi^k = \sum_{i=1}^{n} \lambda_{\phi t_i}^k$ ▷ Party P_k
    Set $\bar{\lambda}_\phi^k = \lambda_\phi^k + Lap(\frac{\Delta_f}{\varepsilon})$ ▷ Party P_k
    Send $\bar{\lambda}_\phi^k$ to server ▷ Party P_k
  end for
  for each $\lambda_\phi^{k*} \in \boldsymbol{\lambda}_\phi^{k*}$ **do**
    Compute $\lambda_\phi^{k*} = \sum_{i=1}^{n} \lambda_{\phi t_i}^{k*}$ using secure vector multiplication ▷ Party P_k, P_*
    Set $\bar{\lambda}_\phi^{k*} = \lambda_\phi^{k*} + Lap(\frac{\Delta_f}{\varepsilon})$ ▷ Party P_k
    Send $\bar{\lambda}_\phi^{k*}$ to server ▷ Party P_k
  end for
end for
Let $\bar{f}_D(\mathbf{w}) = \sum_{j=0}^{J} \sum_{\phi \in \mathbf{\Phi}_j} \left[\bar{\lambda}_\phi^k \phi(\mathbf{w}^k) + \bar{\lambda}_\phi^{k*} \phi(\mathbf{w}^k, \mathbf{w}^*)\right]$ ▷ Server
Compute $\bar{\mathbf{w}} = \arg\min_{\mathbf{w}} \bar{f}_D(\mathbf{w})$ ▷ Server
Return $\bar{\mathbf{w}}$ ▷ Server

For the cross-party sub-function,

$$h_{D^k D^*} = \sum_{j=0}^{J} \sum_{\phi \in \mathbf{\Phi}_j^{k*}} \lambda_\phi^{k*} \phi(\mathbf{w}^k, \mathbf{w}^*).$$

We use λ_ϕ^k to denote the single-party coefficient and λ_ϕ^{k*} to denote the cross-party coefficient. Note that $\phi(\mathbf{w}^k, \mathbf{w}^*)$ can be the first order parameter w^k that needs label from P_1, the second order cross-party parameter $w^k \cdot w^l$ or higher order parameter that involves more than two parties.

After dissecting the objective functions, there are two types of polynomial coefficients required for the server to obtain the overall objective function, i.e. single-party coefficients $\{\boldsymbol{\lambda}_\phi^k\}^K$ and cross-party coefficients $\{\boldsymbol{\lambda}_\phi^{k*}\}^K$. Only sub-dataset D^k is required to compute λ_ϕ^k. The cross-party computation using multiple sub-datasets D^k and D^* is required to compute λ_ϕ^{k*}. The server requests single-party coefficients $\boldsymbol{\lambda}_\phi^k$ from party P_k and cross-party coefficients $\boldsymbol{\lambda}_\phi^{k*}$ from party P_k and P_*. The allocation of these single-party and cross-party coefficients is different to each party. It depends on the type of model, the order of parameters and the availability of label.

Take linear regression for an example. The objective function of linear regression $f_D(\mathbf{w})$ is defined as Equation 2. The server dissects the objective function among the parties and sends the formula of coefficients to inform the parties what they need to compute.

The single-party sub-function for the label owner (party P_1) is

$$g_{D^1} = \sum_{i=1}^{n} (y_i)^2 + \left(-2\sum_{i=1}^{n} y_i \mathbf{x}_i^{1T}\right) \mathbf{w}^1 + \sum_{i=1}^{n} \left(\mathbf{x}_i^1\right)^2 \circ \left(\mathbf{w}^1\right)^2,$$

thus $\boldsymbol{\lambda}_\phi^1 = \{\lambda_{\phi_0}^1, \boldsymbol{\lambda}_{\mathbf{w}^1}^1, \boldsymbol{\lambda}_{(\mathbf{w}^1)^2}^1\}$.

The single-party sub-function for each party $P_k (k \neq 1)$ is

$$g_{D^k} = \sum_{i=1}^{n} \left(\mathbf{x}_i^1\right)^2,$$

thus $\boldsymbol{\lambda}_\phi^k = \{\boldsymbol{\lambda}_{(\mathbf{w}^k)^2}^k\}$.

The cross-party sub-function for all the pairs of party P_k and P_* is

$$\sum_{1 \leq k,* \leq K} h_{D^k D^*} = \sum_{k=1}^{K} \left(-2\sum_{i=1}^{n} y_i \mathbf{x}_i^{kT} \mathbf{w}^k\right) + \sum_{1 \leq k,l \leq K} \left(\sum_{i=1}^{n} (\mathbf{x}_i^k \cdot \mathbf{x}_i^l) \circ (\mathbf{w}^k \cdot \mathbf{w}^l)\right),$$

thus $\{\boldsymbol{\lambda}_\phi^{k*}\}^K = \{\boldsymbol{\lambda}_{\mathbf{w}^k}^{1k}, \boldsymbol{\lambda}_{\mathbf{w}^k \cdot \mathbf{w}^l}^{kl}\}^K$.

We can also look at it based on different orders of polynomial coefficients. All $(1+d+d^2)$ coefficients in Equation 2 are allocated as follow. For the zero-order coefficient λ_{ϕ_0}, it only needs party P_1. For the first order coefficients $\boldsymbol{\lambda}_{\mathbf{\Phi}_1}$, $\boldsymbol{\lambda}_{\mathbf{w}^1}^1$ (of size d^1) from party P_1 are single-party coefficients, and $\boldsymbol{\lambda}_{\mathbf{w}^k}^{1k}$ (of size d^k) for each party $P_k (k \neq 1)$ require communicated information between P_1 and P_k because P_k does not own the label and need y_i from P_1. For the second order coefficients $\boldsymbol{\lambda}_{\mathbf{\Phi}_2}$, all $\boldsymbol{\lambda}_{(\mathbf{w}^k)^2}^k$ (of size $(d^1)^2$) are single-party coefficients for each party P_k, and $\boldsymbol{\lambda}_{\mathbf{w}^k \cdot \mathbf{w}^l}^{kl}$ (of size $d^k \cdot d^l$) require information from two parties to compute coefficients of $\mathbf{w}^k \cdot \mathbf{w}^l$ for each pair of parties P_k, P_l.

2) Collecting the Single-party and Cross-party Coefficients: Before each party P_k provides necessary information to the server, the server decides the scale of noise needed for the model to satisfy ε-differential privacy w.r.t. the whole dataset D simply based on the input space (dimension d and range of $\mathbf{x}_i, y_i$). To achieve ε-differential privacy, the functional mechanism adds $Lap(\frac{\Delta_f}{\varepsilon})$ noise to the polynomial coefficients of the objective function. The server calculates the global sensitivity Δ_f of the objective function $f_D(\mathbf{w})$, and then informs the parties the scale of noise that the parties need to add when sending the results to the server.

Lemma 1. *The global sensitivity of $f_D(\mathbf{w})$ is:*

$$\Delta_f = 2\max_t \sum_{j=1}^{J} \sum_{\phi \in \mathbf{\Phi}_j} ||\lambda_{\phi t}||_1. \tag{6}$$

Because the sensitivity considers the worst case in the input space, the server can calculate the scale of noise needed for global model without getting any raw data from local parties.

After the parties receive the coefficients they need to compute and the amount of noise they need to add onto the results, each party adds Laplace noise $Lap(\frac{\Delta_f}{\varepsilon})$ to both single-party coefficients $\boldsymbol{\lambda}_\phi^k$ and cross-party coefficients $\boldsymbol{\lambda}_\phi^{k*}$, and then sends noisy coefficients $\bar{\boldsymbol{\lambda}}_\phi^k, \bar{\boldsymbol{\lambda}}_\phi^{k*}$ to server.

3) Secure Vector Multiplication: When cross-party communication is needed, parties will not share detail data unless through secure multiparty computation methods. Because each cross-party coefficient $\lambda_\phi^{k*} = \sum_{i=1}^{n} v_i^k \cdot v_i^*$ is a scalar product of two vectors $\mathbf{v}^k, \mathbf{v}^*$ from party P_k and P_*, the only secure operation required to compute $\boldsymbol{\lambda}_\phi^{k*}$ is the scalar product of two vectors.

We run the MPC protocol based on Shamir's secret sharing to handle multiparty secure vector multiplication. Each party sends the encrypted vector to the server. The server computes all scalar products without actually knowing any information in the vectors. The participating parties receive the encrypted results back and jointly decrypt the actual results. The server has zero-knowledge on the raw data during the secure vector multiplication process. Then the parties add Laplace noise $Lap(\frac{\Delta_f}{\varepsilon})$ to these cross-party coefficients and send the noisy results to server.

The total number of operations of secure vector multiplication is $O(d^J)$, and all operations occur one and only one round at the beginning of our approach. Take linear regression for an example, if the features are evenly distributed among parties, the total number of operations of secure vector multiplication is $\frac{K-1}{K}d + \binom{K}{2}\left(\frac{d}{K}\right)^2 = \left(1-\frac{1}{K}\right)\left(d+\frac{d^2}{2}\right)$. For other approaches using secure aggregation scheme to update gradients [19], the number of operations of secure vector multiplication increases by at least a magnitude of the number of iterations.

Theorem 1. *Algorithm 1 satisfies ε-differential privacy w.r.t. the whole dataset D.*

Proof. Assume D and D' are two neighbouring datasets. Without loss of generality, D and D' differ in exact one row t_r and t'_r. The global sensitivity Δ_f is calculated by Equation 6. We have

$$\begin{aligned}
\frac{\Pr\{\bar{f}(\mathbf{w})|D\}}{\Pr\{\bar{f}(\mathbf{w})|D'\}} &= \frac{\prod_{j=1}^{J}\prod_{\phi\in\mathbf{\Phi}_j}\exp\left(\frac{\varepsilon\left|\left|\sum_{t_i\in D}\lambda_{\phi t_i}-\bar{\lambda}_\phi\right|\right|_1}{\Delta_f}\right)}{\prod_{j=1}^{J}\prod_{\phi\in\mathbf{\Phi}_j}\exp\left(\frac{\varepsilon\left|\left|\sum_{t'_i\in D'}\lambda_{\phi t'_i}-\bar{\lambda}_\phi\right|\right|_1}{\Delta_f}\right)} \\
&\le \prod_{j=1}^{J}\prod_{\phi\in\mathbf{\Phi}_j}\exp\left(\frac{\varepsilon}{\Delta_f}\cdot\left|\left|\sum_{t_i\in D}\lambda_{\phi t_i}-\sum_{t'_i\in D'}\lambda_{\phi t'_i}\right|\right|_1\right) \\
&= \prod_{j=1}^{J}\prod_{\phi\in\mathbf{\Phi}_j}\exp\left(\frac{\varepsilon}{\Delta_f}\cdot\left|\left|\lambda_{\phi t_r}-\lambda_{\phi t'_r}\right|\right|_1\right) \\
&= \exp\left(\frac{\varepsilon}{\Delta_f}\cdot\sum_{j=1}^{J}\sum_{\phi\in\mathbf{\Phi}_j}\left|\left|\lambda_{\phi t_r}-\lambda_{\phi t'_r}\right|\right|_1\right) \\
&\le \exp\left(\frac{\varepsilon}{\Delta_f}\cdot 2\max_t\sum_{j=1}^{J}\sum_{\phi\in\mathbf{\Phi}_j}||\lambda_{\phi t}||_1\right) = \exp(\varepsilon).
\end{aligned}$$

□

In Theorem 1, Algorithm 1 satisfies ε-differential privacy w.r.t. the whole dataset D, which is the same as the centralized scenario. In comparison to the centralized functional mechanism, the proposed framework adds the same amount of noise to achieve ε-differential privacy and uses secure vector multiplication to achieve the same utility. In comparison to the methods that add noise onto gradients for each iteration, our framework only needs one round of noise addition and one round of secure multiparty computation.

Claim 1. *Algorithm 1 achieves the same utility in the multiparty setting in comparison to the centralized setting. The utility of Algorithm 1 does not change along with the number of participating parties K.*

4) Differential Privacy for Local Parties: Each party P_k cares about all the coefficients that involve D^k, i.e. the single-party coefficients for party P_k $\boldsymbol{\lambda}_\phi^k$ in the sub-function g and the cross-party coefficients involving party P_k $\boldsymbol{\lambda}_\phi^{k*}$ in the sub-function h.

Lemma 2. *The sensitivity of $f_D(\mathbf{w})$ w.r.t. the sub-dataset D^k belonging to party P_k is*

$$\Delta_f^k = 2\max_t\sum_{j=1}^{J}\sum_{\phi\in\mathbf{\Phi}_j^{(k)}}||\lambda_{\phi t}^{(k)}||_1, \tag{7}$$

where $\lambda_\phi^{(k)}$ indicates either λ_ϕ^k or λ_ϕ^{k*}.

The availability of labels makes significant difference in Δ_f^k. Because only party P_1 owns the label and the label is granted access to other parties, there are more cross-party coefficients for party P_1 than other parties, which increases its corresponding sensitivity Δ_f^1.

Theorem 2. *Algorithm 1 satisfies $\varepsilon^{(k)}$-differential privacy w.r.t. the sub-dataset D^k belonging to party P_k, where $\varepsilon^{(k)} = \frac{\Delta_f^k}{\Delta_f}\varepsilon$.*

Proof. Assume that D^k and $D^{k'}$ are two neighbouring datasets. Without loss of generality, D^k and $D^{k'}$ differ in row t_r and t'_r. The sensitivity Δ_f^k w.r.t. the sub-dataset D^k is calculated by Equation 7. We have

$$\begin{aligned}
\frac{\Pr\{\bar{f}(\mathbf{w})|D^k\}}{\Pr\{\bar{f}(\mathbf{w})|D^{k'}\}} &= \frac{\prod_{j=1}^{J}\prod_{\phi\in\mathbf{\Phi}_j^{(k)}}\exp\left(\frac{\varepsilon\left|\left|\sum_{t_i\in D^k}\lambda_{\phi t_i}^{(k)}-\bar{\lambda}_\phi^{(k)}\right|\right|_1}{\Delta_f}\right)}{\prod_{j=1}^{J}\prod_{\phi\in\mathbf{\Phi}_j^{(k)}}\exp\left(\frac{\varepsilon\left|\left|\sum_{t'_i\in D^{k'}}\lambda_{\phi t'_i}^{(k)}-\bar{\lambda}_\phi^{(k)}\right|\right|_1}{\Delta_f}\right)} \\
&\le \exp\left(\frac{\varepsilon}{\Delta_f}\cdot 2\max_t\sum_{j=1}^{J}\sum_{\phi\in\mathbf{\Phi}_j^{(k)}}||\lambda_{\phi t}^{(k)}||_1\right) = \exp(\frac{\Delta_f^k}{\Delta_f}\varepsilon).
\end{aligned}$$

□

To build an ε-differential privacy global model, the local party P_k can achieve $\varepsilon^{(k)}$-differential privacy, where $\varepsilon^{(k)} = \frac{\Delta_f^k}{\Delta_f}\varepsilon < \varepsilon$, which means stronger privacy guarantee.

Again, take linear regression for an example. To achieve ε-differential privacy w.r.t. the whole dataset D, functional

mechanism adds $Lap(\frac{\Delta_f}{\varepsilon})$ noise to polynomial coefficients. The global sensitivity Δ_f of $f_D(\mathbf{w})$ w.r.t. the whole dataset D is $\Delta_f = 2(1+2d+d^2)$. Party P_1 cares about the coefficients related to D^1: $\lambda^1_{\phi_0}, \boldsymbol{\lambda}^1_{\mathbf{w}^1}, \boldsymbol{\lambda}^{1k}_{\mathbf{w}^k}, \boldsymbol{\lambda}^1_{(\mathbf{w}^1)^2}$ and $\boldsymbol{\lambda}^{1k}_{\mathbf{w}^1 \cdot \mathbf{w}^k} (k \neq 1)$. So the sensitivity of $f_D(\mathbf{w})$ w.r.t. sub-dataset D^1 is

$$\begin{aligned}\Delta^1_f &= 2(1 + 2d^1 + 2(d - d^1) + (d^1)^2 + d^1(d - d^1)) \\ &= 2(1 + 2d + d^1 d).\end{aligned}$$

For party P_1, Algorithm 1 achieves $(\frac{\Delta^1_f}{\Delta_f}\varepsilon)$-differential privacy w.r.t. D^1. Party $P_k (k \neq 1)$ cares about the coefficients related to D^k: $\boldsymbol{\lambda}^{1k}_{\mathbf{w}^k}, \boldsymbol{\lambda}^k_{(\mathbf{w}^k)^2}$ and $\boldsymbol{\lambda}^{kl}_{\mathbf{w}^k \cdot \mathbf{w}^l}$. So the sensitivity of $f_D(\mathbf{w})$ w.r.t. $D^k (k \neq 1)$ is

$$\Delta^k_f = 2(2d^k + (d^k)^2 + d^k(d - d^k)) = 2(2d^k + d^k d).$$

For party $P_k (k \neq 1)$, Algorithm 1 achieves $(\frac{\Delta^k_f}{\Delta_f}\varepsilon)$-differential privacy w.r.t. D^k.

The interesting observation is that: when party P_1 shares label with other parties, P_1 has $4(d - d^1)$ more sensitivity. Δ^k_f for other parties does not change as the label $|y| \leq 1$. Thus, cross-party communication only costs P_1 extra sensitivity. By sharing the label, party P_1 achieves relatively weaker privacy in comparison to other parties.

C. Application to Logistic Regression

For logistic regression, to achieve ε-differential privacy, the functional mechanism adds $Lap(\frac{\Delta_f}{\varepsilon})$ noise to the polynomial coefficients in Equation 4. More specifically, the first order coefficients $\boldsymbol{\lambda}_{\boldsymbol{\Phi}_1}$ contains single-party coefficients $\boldsymbol{\lambda}^1_{\mathbf{w}^1} = \sum_{i=1}^{n} \big(\frac{f_1^{(1)}(0)}{1!} - y_i\big)\mathbf{x}^1_i$ from party P_1 and cross-party coefficients $\boldsymbol{\lambda}^{1k}_{\mathbf{w}^k} = \sum_{i=1}^{n} \big(\frac{f_1^{(1)}(0)}{1!} - y_i\big)\mathbf{x}^k_i$ where party $P_k (k \neq 1)$ does not own the label and need $(\frac{f_1^{(1)}(0)}{1!} - y_i)$ from party P_1. The second order coefficients $\boldsymbol{\lambda}_{\boldsymbol{\Phi}_2}$ contains single-party coefficients $\boldsymbol{\lambda}^k_{(\mathbf{w}^k)^2} = \sum_{i=1}^{n} \frac{f_1^{(2)}(0)}{2!} \big(\mathbf{x}^k_i\big)^2$ from party P_k and cross-party coefficients $\boldsymbol{\lambda}^{kl}_{\mathbf{w}^k \cdot \mathbf{w}^l} = \sum_{i=1}^{n} \frac{f_1^{(2)}(0)}{2!} \mathbf{x}^k_i \cdot \mathbf{x}^l_i$ from party P_k, P_l.

By applying Algorithm 1, the derived $\bar{\mathbf{w}}$ of the global model satisfies ε-differential privacy w.r.t. the whole dataset D according to Theorem 1.

On the other hand, for party P_1, the sensitivity of $f_D(\mathbf{w})$ w.r.t. the sub-dataset D^1 is $\Delta^1_f = d + \frac{d^1(2d - d^1)}{4}$. For party $P_k (k \neq 1)$, the sensitivity of $f_D(\mathbf{w})$ w.r.t. the sub-dataset D^k is $\Delta^k_f = d^k + \frac{d^k(2d - d^k)}{4}$. According to Theorem 2, Algorithm 1 also achieves $(\frac{\Delta^1_f}{\Delta_f}\varepsilon)$-differential privacy w.r.t. D^1 and $(\frac{\Delta^k_f}{\Delta_f}\varepsilon)$-differential privacy w.r.t. D^k. When party P_1 shares label with other parties, party P_1 has $(d - d^1)$ more sensitivity, and Δ^k_f for other parties does not change as the label $y \in \{0, 1\}$ does not change $|(\frac{f_1^{(1)}(0)}{1!} - y)| = \frac{1}{2}$.

D. Extension to the Bottom-up Case

So far, we have discussed the framework from the top-down case, where the server selects the privacy budget to achieve on the whole dataset and informs the parties the scale of noise based on the global sensitivity of objective function. We can also achieve differential privacy from the bottom-up case, where each party selects their own level of differential privacy $\varepsilon^{(k)}$ that they want to achieve for their sub-dataset and the server adjusts the global privacy budget ε accordingly. In practice, the choice between top-down and bottom-up approaches depends on the agreement between the server and participating parties.

In the bottom-up case, each party P_k splits their privacy budget $\varepsilon^{(k)}$ onto sending single-party coefficients $\boldsymbol{\lambda}^k_\phi$ and cross-party coefficients $\boldsymbol{\lambda}^{k*}_\phi$ separately in a differentially private manner, i.e. $\varepsilon^{(k)} = \varepsilon^k + \sum_{l=1}^{K} \varepsilon^{kl}$, where ε^k is the privacy budget for single-party coefficients $\boldsymbol{\lambda}^k_\phi$, and ε^{kl} is the privacy budget for cross-party coefficients $\boldsymbol{\lambda}^{kl}_\phi \subset \boldsymbol{\lambda}^{k*}_\phi$. Party P_k and P_l jointly decide the value of ε^{kl}. The sensitivity of $\boldsymbol{\lambda}^k_\phi$ is $\Delta^k_g = 2 \max_t \sum_{j=1}^{J} \sum_{\phi \in \boldsymbol{\Phi}^k_j} ||\lambda^k_{\phi t}||_1$. The sensitivity of $\boldsymbol{\lambda}^{kl}_\phi$ is $\Delta^{kl}_h = 2 \max_t \sum_{j=1}^{J} \sum_{\phi \in \boldsymbol{\Phi}^{kl}_j} ||\lambda^{kl}_{\phi t}||_1$.

Corollary 1. *The global model achieves ε-differential privacy w.r.t. the whole dataset D in the bottom-up case, where*

$$\varepsilon = \sum_{k=1}^{K} \frac{\Delta_f}{\Delta^k_g}\varepsilon^k + \sum_{1 \leq k,l \leq K} \frac{\Delta_f}{\Delta^{kl}_h}\varepsilon^{kl}.$$

E. Discussion

The objective of our work is to preserve differential privacy for regression models trained on vertically partitioned data. We have theoretical proofs (Theorems 1 and 2) that our algorithm guarantees to satisfy differential privacy. We protect differential privacy of the whole training data such that attackers cannot derive the presence or absence of any single record (with all feature values and label) in the training data from the released jointly-learnt regression model (as shown in Theorem 1). As training data is vertically split into K parties, we further show in Theorem 2 each party k also achieves differential privacy against attackers w.r.t. its own data D^k.

Our contribution is that we add less noise than state-of-the-art approaches to achieve the same level of differential privacy, e.g., our approach reduces noise addition by a magnitude of the number of total iterations compared with gradient perturbation approaches. Furthermore, the secure vector computation also protects the disclosure of raw data between the server and the participating parties. We use the standard MPC protocol based on Shamir's secret sharing scheme in our framework. The number of inner products in our algorithm is bounded by d^J.

IV. Experiments

We evaluate our proposed framework of achieving differential privacy in vertically partitioned multiparty learning based on functional mechanism (FM) for linear regression and logistic regression.

TABLE I: Mean square error of linear regression on US and Brazil datasets under different privacy budgets ε ($\delta = \frac{1}{n}$ for DPFW)

Data	ε	non-private	DPFW-C	DPFW-2	FM
US	0.1	0.0044±0.0132	0.0912±0.0007	0.1286±0.0154	**0.0101±0.0305**
	1		0.0905±0.0008	0.1334±0.0235	**0.0087±0.0261**
	10		0.0903±0.0008	0.1329±0.0204	**0.0086±0.0259**
Brazil	0.1	0.0044±0.0132	0.0470±0.0007	0.1502±0.0921	**0.0070±0.0230**
	1		0.0454±0.0007	0.1989±0.1390	**0.0045±0.0136**
	10		0.0453±0.0006	0.2013±0.1469	**0.0044±0.0132**

TABLE II: Mean square error of linear regression on synthetic datasets under different sparsity s ($\delta = \frac{1}{n}$ for DPFW, $\varepsilon = 1$)

s	non-private	DPFW-C	DPFW-2	DPFW-4	FM
0.1	0.0033±0.0101	0.5487±0.1907	1.2980±0.4537	2.2719±0.7423	**0.0035±0.0107**
0.5	0.0052±0.0157	40.392±5.132	85.264±10.560	164.95±19.60	**0.0058±0.0174**
1.0	0.0050±0.0154	271.09±27.75	638.66±59.70	1206.4±66.2	**0.0056±0.0172**

A. Experiment Setup

1) Dataset: For linear regression, we evaluate on two real-world census datasets US and Brazil [20] datasets. US has 370,000 records and 14 features and Brazil has 190,000 and 14 features. The decisions in US and Brazil datasets are both income (real number). We also evaluate on three synthetic datasets that are sparse and high dimensional. All three synthetic datasets have 80,000 records and 800 features and their sparsity values are $s = 0.1, 0.5, 1.0$, respectively. The sparsity here is both the ratio of nonzero entries in datasets and the ratio of non-zero ground-truth parameters.

For logistic regression, we evaluate on two real-world census datasets Adult [21] and Dutch [22] datasets. Adult has 45,222 samples and 41 features and Dutch has 60,420 records and 36 features. The label in Adult is income (>50k, ≤50k). The label in Dutch is Occupation (occupation w/ low income, occupation w/ high income). We normalize all the numerical features into $[-1, 1]$ and convert all the categorical features to binary variables using one-hot encoding.

We split each dataset into 80% training data and 20% testing data. We replicate experiment for 10 times and report mean and standard deviation.

2) Baseline: For linear regression, we compare with the non-private linear regression and DPFW [14]. DPFW achieves (ε, δ)-differential privacy and has two versions, DPFW-C in the centralized setting and DPFW-K in the multiparty setting. We evaluate our algorithm on varying privacy budget epsilon, the number of parties K and sparsity s. We specify the number of parties $K = 2, 4$ in our comparison, where we randomly partition the datasets into K parts with each party owning the same number of features.

For logistic regression, we compare with the non-private logistic regression and DPSGD [9]. DPSGD adds Laplace noise onto gradients for each iteration. DPSGD does not apply to multiparty setting.

We evaluate utility of linear regression by mean square error (MSE) and utility of logistic regression by accuracy. For the runtime evaluation, we use a secure multiparty computation package MPyC in Python [23], which is based on Shamir's secret sharing scheme. We use a key size of 128-bit in the experiment. We run our algorithim with the number of parties $K = 2, 4$, and compare its runtime to the runtime of the centralized functional mechanism ($K = 1$). Our experiments were carried out on the Dell PowerEdge C4130 with 2 Nvidia Tesla M10 GPU.

B. Utility

1) Linear Regression: For linear regression, we first evaluate our method on two real-world datasets. Table I shows the results on US and Brazil datasets under different values (0.1, 1, 10) of privacy budget ε. We set $\delta = \frac{1}{n}$ for DPFW. Our FM method satisfies $(\varepsilon, 0)$-differential privacy whereas DPFW satisfies (ε, δ)-differential privacy. So our FM method is more restricted in terms of privacy protection. However, our FM method still significantly outperforms DPFW with much smaller MSE in the settings of all three ε values for both datasets, as shown in Table I. In fact, the utility of our method is very close to the non-private linear regression even when the privacy budget ε is small. For example, our FM achieves the MSE of 0.0070 when $\varepsilon = 0.1$ for Brazil data, which is very close to 0.0044 from non-private linear regression. As we discussed previously, our method achieves the same MSE in the decentralized setting as in the centralized setting.

We then evaluate our method on high dimensional synthetic datasets ($d = 800$). Table II shows the results on synthetic datasets under different sparsity s. We set $\varepsilon = 1$ for FM and $\varepsilon = 1, \delta = \frac{1}{n}$ for DPFW. DPFW is designed to work for high dimensional and sparse data. As shown in Table II, DPFW works well with satisfactory MSE values when $s = 0.1$ but has very poor utility with large MSE when $s = 0.5, 1.0$. On the contrary, our FM method works consistently well across all three datasets as the FM mechanism does not depend on data sparsity. We emphasize even with $s = 0.1$, our FM method incurs much smaller MSE (2 or 3 orders of magnitude less) than DPFW. Moreover, our method preserves strict $(\varepsilon, 0)$-differential privacy while DPFW preserves $(\varepsilon, \frac{1}{n})$-differential privacy. For our method, MSE does not change along with the number of participating parties K. On contrast, DPFW incurs more utility loss as the number of parties increases.

TABLE III: Classification accuracy of logistic regression on Adult and Dutch datasets under different privacy budget ε

Data	ε	non-DP	DPSGD-C	FM
Adult	0.1	0.8368±0.0029	0.6000±0.0738	**0.6412±0.1463**
	1		0.6956±0.0229	**0.7315±0.0379**
	10		0.8023±0.0071	**0.8132±0.0231**
Dutch	0.1	0.8303±0.0040	0.5060±0.0684	**0.5783±0.0572**
	1		0.6867±0.0373	**0.7166±0.0489**
	10		0.8003±0.0182	**0.8105±0.0086**

TABLE IV: Runtime of linear regression on US and Brazil datasets and logistic regression on Adult and Dutch datasets ($\varepsilon = 1$, unit in seconds)

Model	Data	K=1 (centralized)	K=2	K=4
Linear regression	US	0.08	93.1	244.4
	Brazil	0.06	47.4	124.5
Logistic regression	Adult	0.39	122.7	350.7
	Dutch	0.31	106.3	302.5

2) Logistic Regression: For logistic regression, we evaluate our method on two real-world datasets. Table III shows the results on Adult and Dutch datasets under different privacy budget ε. DPSGD adds Laplace noise onto gradients for each iteration, so the total amount of noise added into the model increases proportionally with the number of iterations. On the contrary, our FM only adds noise to the objective function and only adds once. As shown in Table III, the utility of our method is much better than the utility of DPSGD. Moreover, DPSGD cannot apply to the multiparty setting while our method is applicable and independent of K as the gradients cannot be simply partitioned by features. We also would like to point out that, compared to linear regression, the utility of FM is relatively worse as privacy budget decreases. This is because the order-2 Taylor expansion approximation is biased to the original objective function.

C. Runtime

We evaluate our method on varying numbers of parties K for both linear regression and logistic regression on real-world datasets. The number of features owned by a party is d/K when they are evenly distributed. Even though our method achieves the same MSE (or accuracy) in the de-centralized setting as in the centralized setting, the runtime is not the same with different numbers of parties. Secure inner product calculation is the bottleneck of our framework as the noise addition of achieving differential privacy via functional mechanism is insignificant in terms of computation and communication cost. As we discussed previously, the total number of operations of secure vector multiplication in this case is $\frac{K-1}{K}d + \binom{K}{2}\left(\frac{d}{K}\right)^2 = \left(1-\frac{1}{K}\right)\left(d+\frac{d^2}{2}\right)$. Thus, the runtime increases with the number of parties K.

Table IV shows the runtime comparison of our method with $K = 2, 4$ in comparison to the centralized setting ($K = 1$). For example, the execution time of logistic regression on Dutch dataset is 0.31, 106.3, and 302.5 seconds for $K = 1$, $K = 2$, and $K = 4$ respectively. We emphasize the execution time is dominated by the secure vector multiplication, which can be reduced when more efficient protocols are adopted. Furthermore, we emphasize again that our framework only needs one round of noise addition and one round of secure multiparty computation and achieves the same utility as the centralized functional mechanism. On the contrary, differential privacy preserving methods that add noise onto gradients for each iteration suffer significant utility loss as shown in Section IV-B.

V. Conclusions and Future Work

In this paper, we proposed a new framework for differential privacy preserving multiparty learning in the vertically partitioned setting based on the functional mechanism. Our core idea is based on the functional mechanism that achieves differential privacy of the released model by adding noise to the objective function. In the framework, the server dissects the objective function into single-party and cross-party sub-functions and rewrites them in the polynomial form. For the coefficients in the polynomial form related to one single party, they can be calculated by each party. For those coefficients related to two or multiple parties, we apply secure vector multiplication. To achieve differential privacy, the parties add noise to the coefficients according to global sensitivity and send noisy coefficients back to server. The server then solves the perturbed objective function and releases the private model. Our method needs only one round of noise addition and secure aggregation. The released model in our framework achieves the same utility as applying the functional mechanism in the centralized setting. We evaluate our method on real-world and synthetic datasets for linear and logistic regressions. The experiment results show the effectiveness of our proposed method.

In our framework, we proposed the use of the MPC protocol based on Shamir's secret sharing and in particular chose the secure multiparty computation package MPyC in Python [23] for secure inner product calculation in our evaluation. Secure calculation is the bottleneck of our framework as the noise addition of achieving differential privacy via functional mechanism is insignificant in terms of computation and communication cost. In our experiment, we mainly evaluated accuracy of regression models on varying numbers of parties K (the number of features owned by a party when evenly distributed is d/K). Our theoretical analysis also showed that our algorithm can achieve the same accuracy as the centralized private model regardless of the number of parties. For execution time, we only evaluated the scenarios of up to 4 parties. In our future work, we will conduct comprehensive evaluation of privacy-accuracy tradeoff and execution time due to the change of the number of features d and the number of parties K. We will also explore newly developed secure multiparty computation protocols [24] in our framework.

Acknowledgements

This work was supported in part by NSF 1502273, 1920920, 1937010 and 1946391.

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, 2017, pp. 1273–1282.

[2] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM TIST*, vol. 10, no. 2, pp. 12:1–12:19, 2019.

[3] C. Dwork, F. McSherry, K. Nissim, and A. D. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography, Third*, 2006, pp. 265–284.

[4] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *J. Mach. Learn. Res.*, vol. 12, pp. 1069–1109, 2011.

[5] C. Dwork, "A firm foundation for private data analysis," *Commun. ACM*, vol. 54, no. 1, pp. 86–95, 2011.

[6] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, 2007, pp. 94–103.

[7] K. Nissim, S. Raskhodnikova, and A. D. Smith, "Smooth sensitivity and sampling in private data analysis," in *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, 2007, pp. 75–84.

[8] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, 2016, pp. 308–318.

[9] S. Song, K. Chaudhuri, and A. D. Sarwate, "Stochastic gradient descent with differentially private updates," in *IEEE Global Conference on Signal and Information Processing, GlobalSIP 2013, Austin, TX, USA, December 3-5, 2013*, 2013, pp. 245–248.

[10] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett, "Functional mechanism: regression analysis under differential privacy," *PVLDB*, vol. 5, no. 11, pp. 1364–1375, 2012.

[11] N. Phan, Y. Wang, X. Wu, and D. Dou, "Differential privacy preservation for deep auto-encoders: an application of human behavior prediction," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, 2016, pp. 1309–1316.

[12] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15, 2015, pp. 1310–1321.

[13] C. Heinze-Deml, B. McWilliams, and N. Meinshausen, "Preserving differential privacy between features in distributed estimation," *CoRR*, vol. abs/1703.00403, 2017.

[14] J. Lou and Y. Cheung, "Uplink communication efficient differentially private sparse optimization with feature-wise distributed data," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 2018, pp. 125–133.

[15] Q. Yao, X. Guo, J. T. Kwok, W. Tu, Y. Chen, W. Dai, and Q. Yang, "Privacy-preserving stacking with application to cross-organizational diabetes prediction," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, 2019, pp. 4114–4120.

[16] W. Rudin, *Principles of mathematical analysis*, ser. International series in pure and applied mathematics. McGraw-Hill, 1953.

[17] D. Evans, V. Kolesnikov, and M. Rosulek, "A pragmatic introduction to secure multi-party computation," *Found. Trends Priv. Secur.*, vol. 2, no. 2-3, pp. 70–246, 2018.

[18] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[19] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," *CoRR*, vol. abs/1711.10677, 2017.

[20] IPUMS, "Integrated public use microdata series, international: Version 7.2," 2009. [Online]. Available: https://international.ipums.org

[21] D. Dheeru and E. K. Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[22] I. Zliobaite, F. Kamiran, and T. Calders, "Handling conditional discrimination," in *11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011*, 2011, pp. 992–1001.

[23] B. Schoenmakers, "MPyC: Secure Multiparty Computation in Python," in *the Theory and Practice of Multiparty Computation (TPMPC) 2018 workshop, Aarhus, Denmark, 2018*.

[24] M. Hastings, B. Hemenway, D. Noble, and S. Zdancewic, " Sok: General purpose compilers for secure multi-party computation," In *2019 IEEE symposium on security and privacy (SP)*, pp. 1220-1237.