Privacy-Preserving Distributed Learning via Obfuscated Stochastic Gradients

Shripad Gade [1] and Nitin H. Vaidya [2]

Abstract— **Distributed (federated) learning has become a popular paradigm in recent years. In this scenario private data is stored among several machines (possibly cellular or mobile devices). These machines collaboratively solve a distributed optimization problem, using private data, to learn predictive models. The aggressive use of distributed learning to solve problems involving sensitive data has resulted in privacy concerns. In this paper we present a synchronous distributed stochastic gradient descent based algorithm that introduces privacy via gradient obfuscation in client-server model. We prove the correctness of our algorithm. We also show that obfuscation of gradients via additive and multiplicative perturbations provides privacy to local data against honest-but-curious adversaries.**

I. Introduction

Machine learning has recently found applications in increasingly many fields ranging from personal preference predictions to finance and healthcare. For instance, banks and financial institutions can use persons' financial records and transaction history, to decide if a certain transaction is fraudulent, or to decide if they should increase the credit limits for a certain individual. Similarly hospitals and insurance providers use medical records to predict if re-hospitalization will be needed for specific patient and to model individual risk to certain disorders. Personal financial and medical records are highly sensitive and their privacy needs to be protected. As machine learning comes into common use it is becoming increasingly important to design algorithms and systems that will prioritize privacy considerations. In this work we present a synchronous distributed learning and optimization algorithm with privacy properties.

In a distributed learning scenario, the data is segregated among several machines, possibly mobile devices [1]. Clients compute updates based on local data and iteratively improve the model. This quest to find the best predictive model, essentially implies solving the following optimization problem,

$$\text{Find } x^* \in \arg\min \sum_{i=1}^{C} f_i(x),$$

where, $f_i(x)$ is the local objective function (loss function) of a client i, known only to client i ($1 \leq i \leq C$). The vector x parameterizes the predictive model and x^* represents the best "model parameters" (also referred to as "state"). Observe that often the data stored at a client is large and gradient update computation is very expensive. In such a scenario, one may replace the gradient with an unbiased estimator of the gradient. This is usually done by dividing the dataset into several batches and then randomly picking a batch to represent the dataset for gradient computation. Such an unbiased estimate of gradient is called stochastic gradient and we will be using it here.

Distributed optimization is often attractive due to the reduced communication requirements, since the agents (clients) communicate updates that are often much smaller in size than each agent's local dataset that characterizes its local objective (loss) function. Moreover, distributed methods provide scalability with respect to number of participants and naturally fit the fragmented and segregated data sources [2]–[4].

However, the updates transmitted by the clients often can give strong indications of the dataset (loss function) that generated the update and hence classical optimization algorithms are vulnerable to privacy violations [5].

In this paper we present POLAR-SGD (Private Optimization and Learning Algorithm - Stochastic Gradient Descent) that protects privacy by obfuscating the updates transmitted by the clients. In particular, we use correlated additive and multiplicative perturbations to obfuscate the stochastic gradient provided by clients. Introduction of perturbation helps clients improve the privacy of their information (as elaborated later in the paper) while the correlated nature of perturbations helps us maintain correctness.

Related Work: Several distributed optimization algorithms have been introduced in literature in recent years, including Sub-gradient Descent [6], [7], Dual Averaging [8], Incremental Algorithms [9], [10], Accelerated Gradient [11], ADMM [12] and EXTRA [13]. Convex distributed optimization has been studied in a variety of scenarios involving directed graphs [14], [15], communication link failures and losses [16], asynchronous communication models [17], [18], and stochastic objective functions [7].

Several privacy-preserving optimization algorithms have been proposed in recent years. These methods can be broadly classified into cryptographic and non-cryptographic methods [19]. Cryptographic methods use cryptographic protocols to provide privacy [20]. Partially homomorphic encryption based methods have been aggressively explored in recent years [21]–[23]. Cryptographic protocols however incur high computational overheads and may not be suited for iterative algorithms. Differential privacy has emerged to be a popular

[1] (gade3@illinois.edu) is a graduate student in ECE Department and Coordinated Science Laboratory, at University of Illinois Urbana-Champaign. [2] (nitin.vaidya@georgetown.edu) is a Professor with Computer Science Department at Georgetown University. This research is supported in part by NSF awards 1421918 and 1610543, and Toyota InfoTechnology Center. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the funding agencies or the U.S. government.

solution in the past few years [5], [24]–[28]. Differential privacy involves adding special noise (Gaussian, Laplacian etc.) to the information exchanged so as to maximize the accuracy of the information exchanged while minimizing the probability of inferring the exact record that generated the information. Such methods however are bound by a fundamental trade-off between accuracy and privacy. Consequently, differentially private optimization algorithms do not converge to the optimum. Transformation is another popular non-cryptographic methods for obtaining privacy. It involves converting the problem into a new problem via algebraic transformations such that the solution of the new problem is the same as old problem [29], [30]. Time-correlated randomized perturbations are used in [31] and secure-multi party computation based correlated randomization is used in [32]–[34] to solve privacy-preserving distributed optimization.

Contributions: Our contributions are enumerated below-

- *Algorithm:* We present POLAR-SGD (Private Optimization and Learning Algorithm - Stochastic Gradient Descent). It is a synchronous protocol where clients use correlated additive and multiplicative perturbations to obfuscate the stochastic gradient. These obfuscated stochastic gradients are uploaded to multiple parameter servers, who then use consensus iteration and projected stochastic gradient descent to learn predictive models.
- *Correctness and Privacy:* We prove convergence of POLAR-SGD under a few conditions on the perturbations. While other perturbation based algorithms such as differential privacy incur accuracy loss due to privacy, we show that our algorithm solves the optimization problem correctly. We also discuss privacy characteristics of POLAR-SGD.

II. Problem Formulation

We will first review the system architecture, distributed learning problem and some preliminaries on distributed learning in client-server models.

A. System Architecture

System architecture is depicted in Figure 1. Our system consists of S parameter servers (also referred to as "servers") and C clients. As discussed before, client i represents a computer that has access to its private data and corresponding loss function $f_i(x)$. Parameter server J maintains a copy of the model, x^J.

The clients receive latest model parameters from the parameter servers and the clients in turn transmit updates (based on private data) to improve the latest model estimate. The clients can communicate with one or more parameter servers in each iteration. The parameter servers communicate with each other every few iterations. We assume that parameter servers form a fully connected component whenever they wish to exchange models (parameter vector) with each other. We also assume that each client communicates with more than one parameter servers in every iteration.

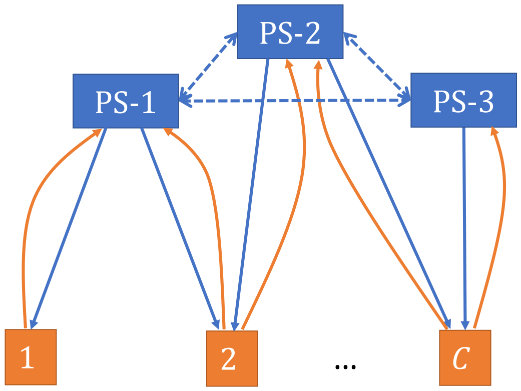

Fig. 1: System Architecture: Parameter Servers (labeled as PS-J) and Clients.

We will assume synchronous and fault-free execution of protocol at all clients and parameter servers. This architecture was analyzed by the authors in [35] and is experimentally validated by Hsieh *et al.* in [36], although [36] does not address privacy as we do.

Observe that the proposed architecture is different from peer-to-peer architecture found commonly in computer networks, sensor networks and robotics. We call our architecture distributed because the private data is owned by local clients and not aggregated. However the gradient updates provided by clients are effectively aggregated when parameter servers perform consensus.

B. Learning Problem

We study the fundamental problem of building a predictive model based on private local data (information). We translate this into a distributed optimization problem as discussed below.

The machine learning model is parameterized as a D-dimensional vector $x \in \mathbb{R}^D$. The set of all feasible model parameters is denoted as $\mathcal{X}$, which is a convex, compact subset of $\mathbb{R}^D$ (i.e. $x \in \mathcal{X} \subset \mathbb{R}^D$).

We assume that the objective function at client i, denoted as $f_i(x)$ is convex and the gradients denoted as $\nabla f_i(x)$ are Lipschitz. We will later show that this assumption can be relaxed for minimum-wait and client-averaged variants of

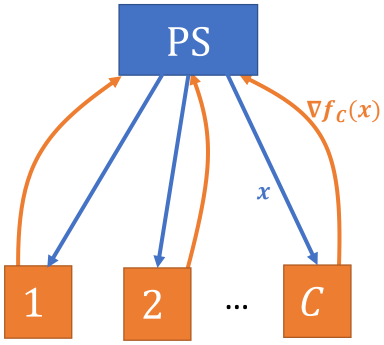

Fig. 2: System with single parameter server [37].

POLAR-SGD which only need sub-gradient of $f_i(x)$ to be bounded. The learning problem is formally presented below.

Problem 1. *Distributed learning implies finding a minimizer to to the objective function* $f(x) \triangleq \sum_{i=1}^{C} f_i(x)$.

$$\textit{Find } x^* \in \arg\min_{x \in \mathcal{X}} f(x)$$

C. Privacy Model

Before we discuss the privacy model, we will review the basic non-privacy preserving algorithm (Figure 2). The algorithm involves iteratively running the following steps -

- Clients download latest model parameters (x) from parameter server and compute gradient of local objective function ($\nabla f_i(x)$).
- Clients upload gradients ($\nabla f_i(x)$) to the server.
- Server performs projected gradient descent using the sum of all received gradients,
  $$x \leftarrow \mathcal{P}_{\mathcal{X}} \left[x - \alpha_k \sum_{i=1}^{C} \nabla f_i(x) \right],$$
  where, α_k is suitably chosen step size and $\mathcal{P}_{\mathcal{X}}$ is the projection operator.

Note that the basic parameter server algorithm above directly exposes local gradients to the parameter server. An honest-but-curious adversary can use these gradients to infer membership of certain data points in the client's private local datasets.

In this work we will assume that multiple parameter servers are available and at most one of the parameter servers is an honest-but-curious adversary. Honest-but-curious adversaries are interested in solving the correct problem and hence they do not tamper with the algorithm (honesty). However, they may run algorithms to try to figure out private information based on observations (curious). Our results can be easily generalized to multiple honest-but-curious adversaries. The parameter servers are often used to provide machine learning as a service. The parameter servers try to learn the best possible model from given data. However, parameter servers may be interested in uncovering private individual data. Our algorithm will protect private data against such honest-but-curious parameter server(s).

D. Notation

Time is indexed as as $\{i, k\}$. Observe that the parameter servers perform consensus periodically. The first index i denotes the number of gradient based descent steps performed since last consensus and the second index refers to the number of consensus operations. We consider consensus to happen after every Δ gradient-based updates. Let $\mathcal{N}_h$ denote the set of servers that can communicate with client h.

III. Algorithm

We reviewed the basic distributed optimization algorithm for parameter server framework [37] in the above section. It can be described as iterative projected gradient descent algorithm where the gradients are collected at a central parameter server. We discuss our algorithm next.

A. POLAR-SGD

Private Optimization and Learning Algorithm - Stochastic Gradient Descent (POLAR-SGD) is a distributed optimization algorithm for multiple parameter-server architecture (Figure 1).

Algorithm Sketch:

1) In POLAR-SGD, clients download latest model parameters from one or more servers and compute stochastic gradients at these values.
2) Clients then obfuscate the stochastic gradients using multiplicative and additive perturbations (detailed later in the section). These obfuscated stochastic gradients are uploaded to the server(s). A client may communicate with any subset of available servers.
3) Each server uses the received stochastic gradients to perform a projected gradient descent step.
4) Servers periodically perform a consensus iteration over the model parameters.

POLAR-SGD algorithm for Clients and Servers is formally presented as Algorithms 1 and 2 respectively.

POLAR-SGD Client: At each iteration $\{i, k\}$, a client h first requests latest model parameters, $x_{i,k}^J$ from each parameter server $J \in \mathcal{N}_h$ (Line 5, Algorithm 1). We consider three variants of POLAR-SGD algorithm. In the first variant (Case 1) called *minimum-wait*, the model parameters that are received first are used by client h and it sets $u = x_{i,k}^I$ where $I \in \mathcal{N}_h$ (Line 6, Algorithm 1). We will discuss two more variants named *client-averaged* (Case 2) and *basic* (Case 3) in next subsection.

Next, each client h then computes stochastic gradient using its private local objective function at the parameter value u (Line 9, Algorithm 1). Note, ξ denotes the data batch used for computing stochastic gradients. Clients then obfuscate the stochastic gradient using additive and multiplicative perturbations. We select these perturbations

Algorithm 1 POLAR-SGD: Client h

```
1:  Input: $x_{i,k}^J$, $\Delta$, NSteps
2:  Result: Upload Obfuscated Gradient - $g_{i,k}^{J,h}$
3:  for k = 1 to NSteps do
4:      for i = 0 to $\Delta$-1 do
5:          Download: $x_{i,k}^J$
6:            Case 1: minimum-wait: $u = x_{i,k}^I$ $(I \in \mathcal{N}_h)$
7:            Case 2: client-averaged: $u = \frac{1}{|\mathcal{N}_h|} \sum_{I \in \mathcal{N}_h} x_{i,k}^I$
8:            Case 3: basic: $u = x_{i,k}^J$
9:          Compute: $g_h(u, \xi) = \nabla f_h(u, \xi)$
10:         Select Perturbation: $W_{i,k}^{J,h}$, $d_{i,k}^{J,h}$
11:         Compute: $g_{i,k}^{J,h} = \left( W_{i,k}[J,h] g_h(u,\xi) + d_{i,k}^{J,h} \right)$
12:         Upload obfuscated stochastic gradient $g_{i,k}^{J,h}$ to
                parameter server $J \in \mathcal{N}_h$.
13:     end for
14: end for
```

(satisfying SLC and BUC) in the Line 10, Algorithm 1 and use them to compute the obfuscated gradient, $g^{J,h}_{i,k}$, in Line 11, Algorithm 1.

$$g^{J,h}_{i,k} = \left(W_{i,k}[J,h] g_h(u,\xi) + d^{J,h}_{i,k}\right) \tag{1}$$

where, $W_{i,k}[J,h]$ is the multiplicative perturbation and $d^{J,h}_{i,k}$ is the additive perturbation assigned by client h to server J at time $\{i,k\}$. For parameter servers $I \notin \mathcal{N}_h$, we have $W_{i,k}[I,h] = 0$ and $d^{I,h}_{i,k} = 0$.

The multiplicative ($W_{i,k}[J,h]$) and additive ($d^{J,h}_{i,k}$) perturbations are arbitrary so long as they satisfy Symmetric Learning and Bounded Update Conditions elaborated below.

Symmetric Learning Condition (SLC) ensures that over a period of Δ gradient descent steps, the multiplicative perturbations assigned by clients are equal and that the additive perturbations sum to zero.

Assumption 1 (SLC). *Equal multiplicative perturbations are assigned to updates from every client over a period of Δ steps. Formally, for each client h,*

$$M \triangleq \sum_{J=1}^{S} \left(\sum_{i=0}^{\Delta-1} W_{i,k}[J,h] \right). \quad (M > 0) \tag{2}$$

Also, the additive perturbations add to zero $\forall\{i,k\}$.

$$\sum_{J=1}^{S} d^{J,h}_{i,k} = 0. \; \forall h \tag{3}$$

Bounded Update Condition (BUC), ensures that the multiplicative and additive perturbations are bounded.

Assumption 2 (BUC). *The sum of absolute value of multiplicative weights over Δ steps is upper bounded by a finite constant ($\bar{M} < \infty$). Formally, for each client h*

$$\sum_{J=1}^{S} \left(\sum_{i=0}^{\Delta-1} |W_{i,k}[J,h]| \right) \leq \bar{M}. \quad (\bar{M} > 0) \tag{4}$$

Also, the additive perturbations are bounded.

$$\|d^{J,h}_{i,k}\| \leq Y. \tag{5}$$

We first observe that each client designs perturbations ($W_{i,k}[J,h]$ and $d^{J,h}_{i,k}$) that satisfy Symmetric Learning Condition (SLC) and Bounded Update Condition (BUC). Since SLC and BUC only involve perturbations that are local to a client, the design of perturbations is local and does not involve coordination with other clients. The finite constants used by clients i.e. M, $\bar{M}$ and Y are known to all clients.

In the next step, Line 12 Algorithm 1, each client h uploads the obfuscated stochastic gradient $g^{J,h}_{i,k}$ to server J.

POLAR-SGD Server: Any parameter server J performs two tasks - 1) using received gradients to perform projected stochastic gradient descent and 2) perform secure consensus over model parameters from all servers.

Algorithm 2 POLAR-SGD: Parameter Server J

```
1: Input: $x^J_{0,1}$, $\alpha_k$, $\Delta$, NSteps
2: Result: $x^* \in \arg\min_{x \in \mathcal{X}} \sum_{i=1}^{C} f_i(x)$
3: for k = 1 to NSteps do
4:     for i = 0 to Δ-1 do
5:         $x^J_{i+1,k} = \mathcal{P}_{\mathcal{X}} \left[ x^J_{i,k} - \alpha_k \sum_{h=1}^{C} g^{J,h}_{i,k} \right]$
6:     end for
7:     Secure Consensus (per coordinate):
8:     Send $R_{J,L} \sim$ U$[0, 2|\mathcal{X}|]$ to each server $L$
9:     Receive $R_{L,J} \sim$ U$[0, 2|\mathcal{X}|]$ from each server $L$
10:    Compute $A^J$
       $A^J = x^J_{\Delta,k} + |\mathcal{X}| + \sum_L R_{L,J} - \sum_L R_{J,L} (mod\ 2|\mathcal{X}|)$
11:    Send and Receive $A^J$
12:    Compute: $x^J_{0,k+1} = \frac{1}{S} \left[ \sum_{I=1}^{S} A^I (mod\ 2|\mathcal{X}|) \right] - |\mathcal{X}|$
13: end for
```

At every iteration $\{i,k\}$, parameter server J uses the obfuscated stochastic gradients $g^{J,h}_{i,k}$ received from clients h at iteration $\{i,k\}$ to perform projected stochastic gradient descent (Line 5, Algorithm 2),

$$x^J_{i+1,k} = \mathcal{P}_{\mathcal{X}} \left(x^J_{i,k} - \alpha_k \sum_{h=1}^{C} g^{J,h}_{i,k} \right) \tag{6}$$

where, α_k is the step size. We assume that $\{\alpha_k\}$ is monotonically non-increasing, positive sequence with $\sum_k \alpha_k = \infty$ and $\sum_k \alpha_k^2 < \infty$.

The servers perform secure consensus over the model parameters (Lines 8-12, Algorithm 2) after every Δ gradient based updates. The secure multi-party computing based private consensus algorithm is from [38]. As discussed in [38], the private consensus algorithm uses *wrap-around* feature of modulo arithmetic and uniformly distributed (U) noise to provide information theoretic privacy while computing sum/average over a fully connected graph. Note that $|\mathcal{X}|$ denotes the maximum value of any coordinate of state $x^J_{i,k}$. Observations of A^J from other servers do not leak information about $x^J_{\Delta,k}$.

B. POLAR-SGD Variants

We described the minimum-wait version of POLAR-SGD, where a client uses the parameter estimates that were received first (Line 6, Algorithm 1). However, there are a couple of more methods for selecting parameter estimates.

1) Client-averaged: All downloaded parameters are averaged and used for gradient computation. Use of average parameter ensures that we only compute one stochastic gradient per iteration (similar to minimum-wait), however, this requires completion of download of parameters from several servers, it has to wait for the slowest server.

$$u = \frac{1}{|\mathcal{N}_h|} \sum_{I \in \mathcal{N}_h} x^I_{i,k}. \tag{7}$$

2) Basic: Clients compute one stochastic gradient per downloaded model parameter. This results in larger computational cost for clients.

$$u = x_{i,k}^J \tag{8}$$

Both *minimum-wait* and *client-averaged* variants of POLAR-SGD lead to efficient and desirable implementations since stochastic gradients are computed only once per iteration. As we will see later in Section IV, since only one stochastic gradient is computed per iteration, we will be able to relax the Lipschitzness condition on gradients, thereby allowing the local objective functions to be non-differentiable.

IV. Main Results

We prove following two results for POLAR-SGD:

- Correctness: algorithm solves distributed optimization problem and asymptotically reaches optimum $x^* \in \mathcal{X}^*$
- Privacy: algorithm does not reveal information about private data via uploaded stochastic gradients

We will first show that the iterates of POLAR-SGD converge to the optimum. Theorem 1 states that any variant of POLAR-SGD, such that the perturbations satisfy SLC and BUC (Assumptions 1,2), converges to the optimum asymptotically with probability 1.

Theorem 1 (Correctness)**.** *Let $f_h(x)$ be convex functions ($\forall h$), and gradients $\nabla f_h(x)$ be bounded and Lipschitz. The iterates generated by any POLAR-SGD variants minimum-wait, client-averaged or basic satisfying SLC and BUC (Assumptions 1, 2) converge to the solution of Problem 1 in $\mathcal{X}^*$ with probability 1.*

In contrast to Theorem 1, the next result shows that even if the functions are not-differentiable, we can still solve the problem as long as the sub-gradients are bounded and we use either minimum-wait or client-averaged case with SLC and BUC satisfied at each $\{i,k\}$. This implies that if $\sum_{J=1}^{S} W_{i,k}[J,h] = M$ and $\sum_{J=1}^{S} |W_{i,k}[J,h]| \leq \bar{M}$ for each client h at any $\{i,k\}$, then the iterates converge to the optimum with probability 1 asymptotically even when $f_h(x)$ are non-differentiable.

Theorem 2. *Let $f_h(x)$ be convex functions ($\forall h$), the sub-gradients $g_h(x)$ be bounded, then the iterates generated by POLAR-SGD variants minimum-wait, or client-averaged satisfying symmetric learning and bounded update condition for each $\{i,k\}$ converge to the solution of Problem 1 in $\mathcal{X}^*$ with probability 1.*

An important take-away of these results should be that we do not trade-off accuracy or correctness of the solution with privacy. The privacy and correctness are kind of independent. The trade-off that we observe is between privacy and the speed of convergence.

We will now discuss the privacy result.

Claim 1. *A honest-but-curious adversarial parameter server J can only observe $g_{i,k}^{J,h}$ from any client h.*

The claim asserts that obfuscated gradients provide improved privacy against honest-but-curious adversaries. Gradient information is observed by parameter server via two sources - 1) gradient estimate received from client and 2) information exchanged between different servers. Obfuscation via additive and multiplicative perturbations protects the gradients from curious parameter servers.

The gradients sent by client h to all other servers can be inferred by an honest-but-curious adversary by observing exchange of information during consensus. Private consensus (adapted from [38]) protects the model parameters from a curious adversary using *wrap-around* feature of modulo arithmetic. The parameters A^J exchanged by servers appear to be uniformly distributed similar to the noise added by agents $\sum_L R_{L,J} - \sum_L R_{J,L}$. The component of gradient received by servers other than J is hence protected against curious adversaries.

V. Analysis and Discussion

A. Convergence Analysis

Let $\mathcal{F}_{i,k}$ denote the $\sigma-$ algebra, $\sigma(\xi_{[i,k]}^h; \forall h)$ generated by the stochastic choice of batches until the time step $\{i,k\}$. Note, $\xi_{[i,k]}^h = (\xi_{0,0}^h, \ldots, \xi_{\Delta,0}^h, \xi_{0,1}^h, \ldots, \xi_{\Delta,1}^h, \ldots \xi_{i,k}^h)$, where $\xi_{i,k}^h$ is the random batch selected by client h at time $\{i,k\}$ in Line 9, Algorithm 1. $\mathcal{F}_{i,k}$ denotes the history of all stochastic gradients uploaded by clients. Clearly $\mathcal{F}_{i,k} \subset \mathcal{F}_{i,k+1}$ from the construction of $\sigma-$algebra.

Observe that the randomness appears from the gradient computation step (batch selection). The multiplicative and additive weights are arbitrary (possibly random) but they *balance-out* due to symmetric learning condition.

Recall the assumptions made in Section II. We assume that $f_i(x)$, the private objective function at client i is convex, the gradients are bounded $\|\nabla f_i(x)\| \leq L$ and Lipschitz, $\|\nabla f_i(x) - \nabla f_i(y)\| \leq N\|x-y\|$ for all i and $x \neq y \in \mathcal{X}$. Moreover we assume the stochastic gradients to be bounded, $\|g_i(u,\xi)\| \leq L$. Next we develop a couple of key lemma's to prove correctness theorems (Theorem 1,2).

We first state a result on convergence of non-negative almost supermartingales (Theorem 1, [39]).

Lemma 2 ([39], [10])**.** *Let $(\Omega, F, \mathcal{P})$ be a probability space and let $F_1 \subset F_2 \subset \ldots$ be a sequence of sub $\sigma-$fields of F. Let u_k, v_k and w_k, $k = 0,1,2,\ldots$ be non-negative F_k-measurable random variables and let $\{\gamma_k\}$ be a deterministic sequence. Assume that $\sum_{k=0}^{\infty} \gamma_k < \infty$ a.s., and $\sum_{k=0}^{\infty} w_k < \infty$ a.s. and*

$$E[u_{k+1}|F_k] \leq (1+\gamma_k)u_k - v_k + w_k,$$

holds with probability 1. Then, the sequence $\{u_k\}$ converges to a non-negative random variable and $\sum_{k=0}^{\infty} v_k < \infty$ almost surely.

The well known non-expansive property (cf. [40]) of Euclidean projection onto a non-empty, closed, convex set $\mathcal{X}$, is represented by the following inequality, $\forall\, x,y \in \mathbb{R}^D$,

$$\|\mathcal{P}_{\mathcal{X}}[x] - \mathcal{P}_{\mathcal{X}}[y]\| \leq \|x-y\|. \tag{9}$$

This non-expansive property of the projection operator will be used extensively in our analysis.

Recall that we assume the parameter servers to form a fully connected topology. However, the correctness results (Theorem 1 and 2) hold for time-varying connected topologies. We will hence assume connected, time-varying topology for the server network.

We model convex averaging using a doubly stochastic matrix[1] to simulate this averaging [6], [41]. At every consensus iteration (k), we assume that the local convex averaging protocol can be written as a doubly stochastic matrix B_k. We will prove convergence when the consensus is performed over incomplete yet connected server graph as this will include the scenario when the parameter servers form a fully connected graph.

We borrow transition matrix analaysis result from [42]. We define transition matrix ($\Phi(k,s)$) as the product of doubly stochastic weight matrices, $\Phi(k,s) = B_k B_{k-1} \dots B_{s+1} B_s, \ (\forall\ k \geq s \geq 0)$. We use analysis from [42] to claim linear convergence of all transition matrix entries $\Phi(k,s)[j,i]$ to $1/S$. That is, $\forall\ k \geq s \geq 0$,

$$\left| \Phi(k,s)[i,j] - \frac{1}{S} \right| \leq \theta \beta^{k-s+1},$$

where, $\theta = (1 - \frac{\rho}{4S^2})^{-2}$ and $\beta = (1 - \frac{\rho}{4S^2})$. depend only on graph topology. Note, ρ is the smallest nonzero entry in matrix B_k for any k.

Disagreement Lemma: First we construct a bound on the disagreement between the parameters ($x^J_{0,k}$) at any server J and the average parameter vector ($\bar{x}_{0,k} = (1/S)\sum_{I=1}^{S} x^I_{0,k}$). This disagreement is denoted by $\delta^J_k = x^J_{0,k} - \bar{x}_{0,k}$.

Lemma 3. *The sequence generated by POLAR-SGD following the Symmetric Learning and Bounded Update conditions,*

$$\max_J \|\delta^J_{k+1}\| \leq S\theta\beta^{k+1} \max_I \|x^I_{0,0}\| + 4\alpha_k \Delta C(\bar{M}L + Y) + 2\theta\Delta SC(\bar{M}L + Y) \sum_{l=1}^{k} \beta^{k+1-l} \alpha_{l-1}$$

We first observe that this is a deterministic bound. We use bounds on the stochastic gradients to compute the bound on $\|\delta^J_{k+1}\|$. The first term shows the geometric decrease of initial disagreement among parameter servers. The other two term reflect the effect of stochastic gradient based updates. Due to space limitations, we will not include proofs in the paper. Detailed proofs can be found in [43] and [35].

Iterate Lemma: The lemma below provides a bound on the distance between the iterates and a point $y \in \mathcal{X}$.

Lemma 4. *The sequence generated by POLAR-SGD following Symmetric Learning and Bounded Update Conditions, satisfies for all $y \in \mathcal{X}$,*

$$\mathbb{E}[\eta^2_{k+1} | \mathcal{F}_{0,k}] = (1 + A_k)\eta^2_k - 2M\alpha_k(f(\bar{x}_{0,k}) - f(y)) + B_k$$

where, $\eta^2_k = (1/S)\sum_{I=1}^{S} \|x^I_{0,k} - y\|^2$,
$A_k = 4\frac{1}{S}\alpha^2_k N\Delta C^2 \bar{M}(\bar{M}L + Y) + 2\frac{1}{S}\alpha_k NC\bar{M} \max_J \|\delta^J_k\|$, *and* $B_k = \alpha^2_k \left(C_1^2 + 4N\Delta C^2 \bar{M}(\bar{M}L + Y)\right) + 2\alpha_k NC\bar{M} \max_J \|\delta^J_k\| + 2\alpha_k C(\bar{M}L + \Delta Y) \max_J \|\delta^J_k\|$.

Note that the expression has similar structure as Lemma 2 and we exploit this to prove convergence. We observe that A_k (respectively γ_k in Lemma 2) is a deterministic sequence and we show that $\sum_k A_k < \infty$. We first prove that $\sum_k \alpha_k \max_J \|\delta^J_k\| < \infty$ a.s. and use it to prove that $\sum_k B_k < \infty$. Invoking Lemma 2 we conclude that $\sum_k \alpha_k(f(\bar{x}_{0,k}) - f(y)) < \infty$ and $\lim_{k\to\infty} \eta_k$ exists. We further use analysis similar to [7] to prove convergence of iterate average to the optimum with probability 1. This proves Theorem 1. The proof for Theorem 2 is very similar to the proof for Theorem 1. We exploit the fact that since only one stochastic gradient is computed by a client and the perturbations that are added and multiplied to the stochastic gradient balance-out due to SLC (at each $\{i,k\}$) and are bounded due to BUC. The sum of these obfuscated gradients is simply the stochastic gradient at any iteration. And the convergence results can be worked out following the intuition.

B. Privacy Analysis

As discussed before there are two possible methods of violating privacy. The first being direct observation of received gradient from a specific client and the second being information about gradient sent to a different server leaking during consensus.

The received gradients are perturbed by multiplicative and additive arbitrary (possibly random) perturbations. The additive perturbation hides gradients especially when the gradient is close to zero (since $W[J,h]g_h(u,\xi)$ will be close to zero too). This protects gradients from direct observation from an adversary. Note that the perturbations are arbitrary and unknown to the adversary, protecting the stochastic gradients. No information leakage happens during consensus as we use secure multi-party computation based private consensus algorithm to average model parameters.

Remark 1. *It is important to note that, while private objective functions owned by individual clients are protected against honest-but-curious adversaries, the overall cost function $f(x)$ is not. Consider a scenario where $\Delta = 1$, where projected descent and consensus steps occur alternatively. As the private consensus step computes the exact average, an adversary can estimate the effect of stochastic gradients on servers other than itself. This along with stochastic gradient directly received from a client, an adversary can estimate the gradient of the whole function, violating the privacy of $f(x)$.*

Remark 1 elaborates the price we pay for exactly solving the given problem. This is common for any correctness preserving algorithm.

Note that we can have the perturbations to be different per coordinate. As long as the symmetric learning and bounded update conditions are satisfied per coordinate, we can easily extend the correctness proofs to per-coordinate weighted obfuscation strategy.

[1] In a fully connected topology, the doubly stochastic matrix is just a matrix with all entries being exactly equal to $1/S$.

C. Extensions and Future Work

We assume that there is a single honest-but-curious adversary. This can be extended to multiple (τ) adversarial parameter servers scenario if clients upload gradients to $\tau+1$ parameter servers. We leave the analysis as future work.

We make an assumption that the parameter servers form a fully connected graph. This assumption allows us to use secure consensus scheme from [38]. However, if the parameter servers are not fully connected but only connected we can claim correctness following the analysis as described above. We will get some privacy under this scenario as well, however its characterization is a work in progress.

As discussed above, it is easy to extend POLAR-SGD to have coordinate wise different multiplicative weights. Under mild additional conditions like per coordinate satisfaction of symmetric learning and bounded update condition we can extend the correctness results.

VI. Experimental Validation

In this section we empirically validate POLAR-SGD. We consider linear regression problem on a large synthetic dataset. We consider 100000 data points partitioned among $C = 100$ clients. Our system consists of $S = 5$ parameter servers. The clients upload gradients to more than one parameter server in each iteration and the servers form a fully connected graph and perform secure consensus ever Δ iterations.

If dataset D_i is stored at client i, then we can write the local objective function at client i as,

$$f_i(x) = \sum_{l \in D_i} \|x^T a_l - b_l\|^2,$$

where, (a_l, b_l) are data points belonging to D_i. And the overall objective function can be written as,

$$f(x) = \sum_i f_i(x) = \sum_{i=1}^{C} \sum_{l \in D_i} \|x^T a_l - b_l\|^2.$$

We consider multiplicative perturbations and additive perturbations that satisfy SLC and BUC with parameters $M = 5$ and $\bar{M} = 50$. We consider three values of $\Delta = \{10, 20, 50\}$ and compare performance. We also compare performance of POLAR-SGD with respect to non-private algorithm and show the trade-off between privacy and convergence speed. Note that we use batch size of 10 samples for computing stochastic gradients in each iteration.

Figure 3 shows the sub-optimality of iterates with respect to iterations. First observe that POLAR-SGD iterates converge (in function value) to the optimum for each Δ. Observe that the non-private algorithm presented earlier converges much faster than POLAR-SGD. As expected, with higher Δ, it takes larger number if iterations for multiplicative and additive perturbations to average out and hence it take longer to converge. The smaller Δ's (like 10 and 20) the iterates follow the general trend of the non-private algorithm very closely. However with larger Δ the function sub-optimality may increase as compared to the starting sub-optimality for few iterations.

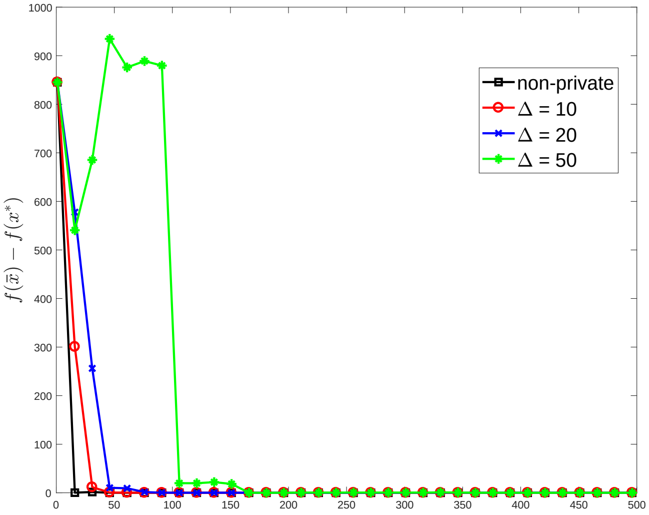

Fig. 3: Sub-optimality v/s iterations.

VII. Conclusion

In this paper we consider privacy preserving distributed optimization algorithm for a multiple parameter server architecture. Our algorithm uses additive and multiplicative weights to perform gradient obfuscation. We show correctness of our algorithm and discuss privacy. We experimentally validate our algorithm on linear regression problem.

References

[1] J. Konečnỳ, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," *arXiv preprint arXiv:1511.03575*, 2015.

[2] T. Kraska, A. Talwalkar, J. C. Duchi, R. Griffith, M. J. Franklin, and M. I. Jordan, "Mlbase: A distributed machine-learning system.," in *CIDR*, vol. 1, pp. 2–1, 2013.

[3] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," in *NIPS*, pp. 19–27, Curran Associates, Inc., 2014.

[4] I. Cano, M. Weimer, D. Mahajan, C. Curino, and G. M. Fumarola, "Towards geo-distributed machine learning," *arXiv preprint arXiv:1603.09035*, 2016.

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

[6] A. Nedić and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *Automatic Control, IEEE Transactions on*, vol. 54, no. 1, pp. 48–61, 2009.

[7] S. S. Ram, A. Nedić, and V. V. Veeravalli, "Distributed stochastic subgradient projection algorithms for convex optimization," *Journal of optimization theory and applications*, vol. 147, no. 3, pp. 516–545, 2010.

[8] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: convergence analysis and network scaling," *IEEE Transactions on Automatic Control*, vol. 57, no. 3, pp. 592–606, 2012.

[9] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pp. 20–27, ACM, 2004.

[10] S. S. Ram, A. Nedić, and V. V. Veeravalli, "Incremental stochastic subgradient algorithms for convex optimization," *SIAM Journal on Optimization*, vol. 20, no. 2, pp. 691–717, 2009.

[11] D. Jakovetić, J. Xavier, and J. M. Moura, "Fast distributed gradient methods," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1131–1146, 2014.

[12] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the admm in decentralized consensus optimization," *IEEE Transactions on Signal Processing*, vol. 62, no. 7, pp. 1750–1761, 2014.

[13] W. Shi, Q. Ling, G. Wu, and W. Yin, "Extra: An exact first-order algorithm for decentralized consensus optimization," *SIAM Journal on Optimization*, vol. 25, no. 2, pp. 944–966, 2015.

[14] A. Nedić and A. Olshevsky, "Distributed Optimization Over Time-Varying Directed Graphs," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 601–615, 2015.

[15] C. Xi and U. A. Khan, "On the linear convergence of distributed optimization over directed graphs," *arXiv:1510.02149*, 2015.

[16] C. N. Hadjicostis, N. H. Vaidya, and A. D. Domínguez-García, "Robust distributed average consensus via exchange of running sums," *IEEE Transactions on Automatic Control*, vol. 61, no. 6, pp. 1492–1507, 2016.

[17] A. Nedić, "Asynchronous broadcast-based convex optimization over a network," *IEEE Transactions on Automatic Control*, vol. 56, no. 6, pp. 1337–1351, 2011.

[18] J. Liu and S. J. Wright, "Asynchronous stochastic coordinate descent: Parallelism and convergence properties," *SIAM Journal on Optimization*, vol. 25, no. 1, pp. 351–376, 2015.

[19] P. C. Weeraddana, G. Athanasiou, C. Fischione, and J. S. Baras, "Per-se privacy preserving solution methods based on optimization," in *Proceedings of the 52nd IEEE Conference on Decision and Control (CDC)*, pp. 206–211, 2013.

[20] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *Security and Privacy (SP), 2017 IEEE Symposium on*, pp. 19–38, IEEE, 2017.

[21] C. Zhang, M. Ahmad, and Y. Wang, "Admm based privacy-preserving decentralized optimization," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 565–580, 2019.

[22] A. B. Alexandru, K. Gatsis, and G. J. Pappas, "Privacy preserving cloud-based quadratic optimization," in *Communication, Control, and Computing (Allerton), 2017 55th Annual Allerton Conference on*, pp. 1168–1175, IEEE, 2017.

[23] A. B. Alexandru, K. Gatsis, Y. Shoukry, S. A. Seshia, P. Tabuada, and G. J. Pappas, "Cloud-based quadratic optimization with partially homomorphic encryption," *arXiv preprint arXiv:1809.02267*, 2018.

[24] J. Hamm, Y. Cao, and M. Belkin, "Learning privately from multiparty data," in *Proceedings of The 33rd International Conference on Machine Learning*, pp. 555–563, 2016.

[25] S. Han, U. Topcu, and G. J. Pappas, "Differentially private distributed constrained optimization," *IEEE Transactions on Automatic Control*, vol. PP, no. 99, pp. 1–1, 2016.

[26] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," *arXiv preprint arXiv:1607.00133*, 2016.

[27] M. Hale and M. Egerstedty, "Differentially private cloud-based multi-agent optimization with constraints," in *2015 American Control Conference (ACC)*, pp. 1235–1240, IEEE, 2015.

[28] M. T. Hale and M. Egerstedt, "Cloud-enabled differentially private multi-agent optimization with constraints," *IEEE Transactions on Control of Network Systems*, 2017.

[29] O. L. Mangasarian, "Privacy-preserving horizontally partitioned linear programs," *Optimization Letters*, vol. 6, no. 3, pp. 431–436, 2012.

[30] C. Wang, K. Ren, and J. Wang, "Secure and practical outsourcing of linear programming in cloud computing," in *INFOCOM, 2011 Proceedings IEEE*, pp. 820–828, IEEE, 2011.

[31] N. E. Manitara and C. N. Hadjicostis, "Privacy-preserving asymptotic average consensus," in *Control Conference (ECC), 2013 European*, pp. 760–765, IEEE, 2013.

[32] S. Gade and N. H. Vaidya, "Private optimization on networks," in *2018 Annual American Control Conference (ACC)*, pp. 1402–1409, IEEE, 2018.

[33] S. Gade and N. H. Vaidya, "Private learning on networks: Part ii," *arXiv preprint arXiv:1703.09185*, 2017.

[34] S. Gade and N. H. Vaidya, "Private learning on networks," *arXiv preprint arXiv:1612.05236*, 2016.

[35] S. Gade and N. H. Vaidya, "Distributed optimization for client-server architecture with negative gradient weights," *arXiv preprint arXiv:1608.03866*, 2016.

[36] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia: Geo-distributed machine learning approaching lan speeds.," in *NSDI*, pp. 629–647, 2017.

[37] M. Li, D. G. Andersen, A. Smola, and K. Yu, "Communication Efficient Distributed Machine Learning with the Parameter Server," in *NIPS*, pp. 1–9, 2014.

[38] E. A. Abbe, A. E. Khandani, and A. W. Lo, "Privacy-preserving methods for sharing financial risk exposures," *The American Economic Review*, vol. 102, no. 3, pp. 65–70, 2012.

[39] H. Robbins and D. Siegmund, "A convergence theorem for non negative almost supermartingales and some applications," in *Herbert Robbins Selected Papers*, pp. 111–135, Springer, 1985.

[40] D. P. Bertsekas, A. Nedić, A. E. Ozdaglar, *et al.*, *Convex analysis and optimization*. Athena Scientific, 2003.

[41] L. Xiao and S. Boyd, "Optimal scaling of a gradient method for distributed resource allocation," *Journal of optimization theory and applications*, vol. 129, no. 3, pp. 469–488, 2006.

[42] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis, "Distributed subgradient methods and quantization effects," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pp. 4177–4184, IEEE, 2008.

[43] S. Gade and N. H. Vaidya, "Privacy-preserving distributed learning via obfuscated stochastic gradients," 2018. (to be uploaded on arXiv).