Private Optimization on Networks

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

***Abstract*— This paper considers a distributed multi-agent optimization problem, with the global objective consisting of the sum of local objective functions of the agents. The agents solve the optimization problem using local computation and communication between adjacent agents in the network. We present two randomized iterative algorithms for distributed optimization. To improve privacy, our algorithms add "structured" randomization to the information exchanged between the agents. We prove deterministic correctness (in every execution) of the proposed algorithms despite the information being perturbed by noise with non-zero mean. We prove that a special case of a proposed algorithm (called function sharing) preserves privacy of individual polynomial objective functions under a suitable connectivity condition on the network topology.**

I. Introduction

Distributed optimization has received a lot of attention in recent years. It involves a system of networked agents that optimize a global objective function $f(x) \triangleq \sum_i f_i(x)$, where $f_i(x)$ is the local objective function of agent i. Each agent is initially only aware of its own local objective function. The agents solve the global optimization problem in an iterative manner. Each agent maintains a "state estimate", which it shares with its neighbors in each iteration, and then updates its state estimate using the information received from the neighbors. A distributed optimization algorithm must ensure that the state estimates maintained by the agents converge to an optimum of the global cost function.

Emergence of networked systems has led to the application of distributed optimization framework in several interesting contexts, such as machine learning, resource allocation and scheduling, and robotics [1], [2]. In a distributed machine learning scenario, partitions of the dataset are stored among several different agents (such as servers or mobile devices [3]), and these agents solve a distributed optimization problem in order to collaboratively learn the most appropriate "model parameters". In this case, $f_i(x)$ at agent i may be a *loss function* computed over the dataset stored at agent i, for a given choice x of the *model parameters* (i.e., here x denotes a vector of model parameters).

Distributed optimization can reduce communication requirements of learning, since the agents communicate information that is often much smaller in size than each agent's local dataset that characterizes its local objective function. The scalability of distributed optimization algorithms, and their applicability for geo-distributed datasets, have made them a desirable choice for distributed learning [4]–[6].

Distributed optimization algorithms rely on exchange of information between agents, making them vulnerable to privacy violations. In particular, in case of distributed learning, the local objective function of each agent is derived using a local dataset known only to that agent. Through the information exchanged between agents, information about an agent's local dataset may become known to other agents. Therefore, privacy concerns have emerged as a critical challenge in distributed optimization [7], [8].

In this paper we present two algorithms that use "structured randomization" of state estimates shared between agents. In particular, our structured randomization approach obfuscates the state estimates by adding *correlated random noise*. Introduction of random noise into the state estimates allows the agents to improve privacy. *Correlation* (as elaborated later) helps to ensure that our algorithms asymptotically converge to a true optimum, despite perturbation of state estimates with non-zero mean noise. We also prove strong privacy guarantees for a special case of our algorithm for a distributed polynomial optimization problem. Contributions of this paper are as follows:

- We present *Randomized State Sharing* (RSS) algorithms for distributed optimization that use structured randomization. Our first algorithm, named RSS-NB, introduces noise that is *Network Balanced* (NB), as elaborated later, and the second algorithm, RSS-LB, introduces *Locally Balanced* (LB) noise. We prove *deterministic* convergence (in every execution) to an optimum, despite the use of randomization.
- We consider a special case of RSS-NB (called "Function Sharing" or FS), where the random perturbations added to local iterates are state-dependent. State-dependent random perturbations simulate the obfuscation of objective function using a *noise function*. We argue that the FS algorithm achieves a strong notion of privacy.
- We use RSS-NB and RSS-LB algorithms to train a deep neural network for digit recognition using the MNIST dataset, and to train a logistic regression model for document classification of the Reuters dataset. The experiments validate our theoretical results and we show that we can obtain high accuracy models, despite introducing randomization to improve privacy.

Related Work: Many distributed optimization algorithms have appeared in the literature over the past decade, including Sub-gradient Descent [9], [10], Dual Averaging [11], Incremental Algorithms [2], [12], Accelerated Gradient

[1] (gade3@illinois.edu) and [2] (nhv@illinois.edu) are with ECE Department, and Coordinated Science Laboratory, at University of Illinois Urbana-Champaign. 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.

[13], ADMM [14] and EXTRA [15]. Solutions to distributed optimization of convex functions have been proposed for myriad scenarios involving directed graphs [16], [17], communication link failures and losses [18], asynchronous communication models [19]–[21], and stochastic objective functions [10], [22].

Privacy-preserving methods for optimization and learning can be broadly classified into cryptographic approaches and non-cryptographic approaches [23]. Cryptography-based privacy preserving optimization algorithms [24], [25] tend to be computationally expensive. Non-cryptographic approaches have gained popularity in recent years. ϵ-*differential privacy* is a probabilistic technique that involves use of randomized perturbations [26]–[30] to minimize the probability of uncovering of specific records from databases. Differential privacy methods, however, suffer from a fundamental tradeoff between the accuracy of the solution and the privacy margin (parameter ϵ) [29]. *Transformation* is another non-cryptographic technique that involves converting a given optimization problem into a new problem via algebraic transformations such that the solution of the new problem is the same as the solution of the old problem [31], [32]. This enables agents to conceal private data effectively while the quality of solution is preserved. Transformation approaches in literature, however, cater only to a relatively small set of problem classes.

II. Notation and Problem Formulation

We consider a *synchronous* system consisting of n agents connected using a network of undirected (i.e., bidirectional) communication links. The communication links are always reliable. The set of agents is denoted by $\mathcal{V}$; thus, $|\mathcal{V}| = n$.

Although all the links are undirected, for convenience, we represent each undirected link using a pair of directed edges. Define $\mathcal{E}$ as a set of directed edges corresponding to the communication links in the network:

$$\mathcal{E} = \{(u, v) : u, v \in \mathcal{V} \text{ and } u \text{ communicates with } v\}.$$

Thus, the communication network is represented using a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The neighbor set of agent v is defined as the set of agents that are connected to agent v. By convention, $\mathcal{N}_v$ includes v itself, i.e. $\mathcal{N}_v = \{u \,|(u, v) \in \mathcal{E}\} \cup \{v\}$.

We assume that the communication graph $\mathcal{G}$ is time-varying, however, it is connected at every iteration. We impose additional connectivity constraint later when discussing privacy in Section IV.

The focus of this paper is on iterative algorithms for distributed optimization. Each agent maintains a *state estimate*, which is updated in each iteration of the algorithm. The state estimate at agent i at the start of iteration k is denoted by x_k^i. We assume that argument x of $f_i(x)$ is constrained to be in a feasible set $\mathcal{X} \subset \mathbb{R}^D$. The state estimate of each agent is initialized to an arbitrary vector in $\mathcal{X}$. For $z \in \mathbb{R}^D$, we define projection operator $\mathcal{P}_\mathcal{X}$ as, $\mathcal{P}_\mathcal{X}(z) = \arg\min_{y \in \mathcal{X}} \|z - y\|$. Problem 1 below formally defines the goal of distributed optimization.

Problem 1. *Given local objective function $f_i(x)$ at each agent $i \in \mathcal{V}$, and feasible set $\mathcal{X} \subset \mathbb{R}^D$ (i.e., the set of feasible x), design a distributed algorithm such that, for some*

$$x^* \in \arg\min_{x \in \mathcal{X}} \sum_{i=1}^{n} f_i(x),$$

we have $\lim_{k \to \infty} \; x_k^i = x^*, \; \forall i \in \mathcal{V}$.

Let f^* denote the optimal value of $f(x)$, and let $\mathcal{X}^*$ denote the set of all optima of $f(x)$, i.e.

$$f^* = \inf_{x \in \mathcal{X}} f(x) \text{ and } \mathcal{X}^* = \{x \mid x \in \mathcal{X}, f(x) = f^*\}.$$

Let $\|.\|$ denote the Euclidean norm. We make the following assumptions.

Assumption 1 (Objective Function and Feasible Set)**.**

1) *The feasible set, $\mathcal{X}$, is a non-empty, convex, and compact subset of $\mathbb{R}^D$.*
2) *The objective function $f_i : \mathcal{X} \to \mathbb{R}, \forall i \in \mathcal{V}$, is a convex function. Thus, $f(x) := \sum_{i=1}^n f_i(x)$ is also a convex function.*
3) *The set of optima, $\mathcal{X}^*$, is non-empty and bounded.*

Assumption 2 (Gradient Bound and Lipschitzness [1])**.**

1) *The gradients are norm-bounded, i.e., $\exists \; L > 0$ such that, $\|\nabla f_i(x)\| \leq L, \; \forall \; x \in \mathcal{X}$ and $\forall \; i \in \mathcal{V}$.*
2) *The gradients are Lipschitz continuous, i.e., $\exists \; N > 0$ such that, $\|\nabla f_i(x) - \nabla f_i(y)\| \leq N\|x - y\|, \; \forall \; x, y \in \mathcal{X}$ and $\forall \; i \in \mathcal{V}$.*

III. Distributed Algorithms

This section first presents an iterative Distributed Gradient Descent algorithm (DGD) from prior literature [9]. Later we modify DGD to improve privacy. In particular, we present two algorithms based on *Randomized State Sharing* (RSS).

A. DGD *Algorithm [9]*

Iterative distributed algorithms such as Distributed Gradient Descent (DGD) use a combination of consensus dynamics and local gradient descent to distributedly find a minimizer of $f(x)$. More precisely, in each iteration, each agent receives state estimates from its neighbors and performs a consensus step followed by descent along the direction of the gradient of its local objective function.

The pseudo-code for the DGD algorithm is presented below as Algorithm 0. The algorithm presents the steps performed by any agent $j \in \mathcal{V}$. The different agents perform their steps in parallel. Lines 4-5 are intentionally left blank in Algorithm 0, to facilitate comparison with other algorithms presented later in the paper.

As shown on Lines 6 and 7 of Algorithm 0, in the k-th iteration, each agent j first sends its current estimate x_k^j to the neighbors, and then receives the estimates from all its neighbors. Using these estimates, as shown on line 8, each agent performs a *consensus step* (also called *information*

[1]The compactness of $\mathcal{X}$ along with Lipschitzness of Gradient (Assumption 2-2) implies gradient boundedness (Assumption 2-1).

Algorithm 0 DGD Algorithm [9]

1: Input: α_k $(k \geq 1)$, MAX_ITER.
Initialization: $x_0^j \in \mathcal{X},\ \forall j \in \mathcal{V}$.
2: The steps performed by each agent $j \in \mathcal{V}$:
3: **for** k = 1 to MAX_ITER **do**
4:
5:
6: Send estimate x_k^j to each agent $i \in \mathcal{N}_j$
7: Receive estimate x_k^i from each agent $i \in \mathcal{N}_j$
8: Information Fusion:
$v_k^j = \sum_{i \in \mathcal{N}_j} B_k[j,i] x_k^i$
9: Projected Gradient Descent:
$x_{k+1}^j = \mathcal{P}_{\mathcal{X}} \left[v_k^j - \alpha_k \nabla f_j(v_k^j) \right]$
10: **end for**

fusion), which involves computing a convex combination of the state estimates. The resulting convex combination is named v_k^j. Matrix B_k used in this step is a doubly stochastic matrix [9], which can be constructed by the agents using previously proposed techniques,[2] such as Metropolis weights [33]. The Metropolis weights are:

$$B_k[i,j] = \begin{cases} 1/(1+\max(|\mathcal{N}_i|, |\mathcal{N}_j|)) & \text{if } j \in \mathcal{N}_i \\ 1 - \sum_{l \neq i} B_k[i,l] & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Agent j performs *projected gradient descent* step (Line 9, Algorithm 0) involving descent from v_k^j along the local objective function's gradient $\nabla f_j(v_k^j)$, followed by projection onto the feasible set $\mathcal{X}$. This step yields the new state estimate at agent j, namely, x_{k+1}^j. α_k used on line 9 is called the *step size*. The sequence α_k, $k \geq 1$, is a non-increasing sequence such that $\sum_{k=1}^{\infty} \alpha_k = \infty$ and $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$.

Prior work [9] has shown that DGD Algorithm 0 solves Problem 1, that is, the agents' state estimates asymptotically reach consensus on an optimum in $\mathcal{X}^*$.

DGD is not designed to be privacy-preserving and an adversary may learn information about an agent's local objective function by observing information exchange between the agents. We now introduce algorithms that *perturb* the state estimates before the estimates are shared between the agents. The perturbations are intended to hide the true state estimate values and improve privacy.

B. RSS-NB *Algorithm*

The first proposed algorithm, named *Randomized State Sharing−Network Balanced* (RSS-NB) is a modified version of Algorithm 0. The pseudo-code for algorithm RSS-NB is presented as Algorithm 1 below.

Random variables $s_k^{i,j}$ are used to compute the perturbations. Any sampling strategy would work for generating $s_k^{i,j}$ as long as they are norm bounded (as detailed later in the section). We will discuss the procedure for computing the perturbation after describing the rest of the algorithm. As we will discuss in more detail later, on Line 4 of Algorithm 1, agent j computes perturbation d_k^j to be used in iteration k. On Line 4, the perturbation is weighted by step size α_k and added to state estimate x_k^j to obtain the perturbed state estimate w_k^j of agent j. That is,

$$w_k^j = x_k^j + \alpha_k d_k^j. \tag{1}$$

α_k here is the step size, which is also used in the information fusion step in Line 8. Properties satisfied by α_k are identical to those in the DGD Algorithm 0.

Having computed the perturbed estimate w_k^j, each agent then sends the perturbed estimate w_k^j to its neighbors (Line 6) and receives the perturbed estimates of the neighbors (Line 7, Algorithm 1). Similar to Algorithm 0, Steps 8 and 9 of the RSS-NB algorithm also perform information fusion using a doubly stochastic matrix B_k, followed by projected gradient descent.

Now we describe how the perturbation $d_k^j \in \mathbb{R}^D$ is computed on Line 4 of Algorithm 1. The strategy for computing the perturbation is motivated by a secure distributed averaging algorithm in [34]. In iteration k, the computation of the perturbation d_k^j at agent j uses variables $s_k^{j,i}$ and $s_k^{i,j}$, $i \in \mathcal{N}_j$, which take values in $\mathbb{R}^D$. As shown on Line 4, the perturbation d_k^j is computed as follows.

$$d_k^j = \sum_{i \in \mathcal{N}_j} s_k^{i,j} - \sum_{i \in \mathcal{N}_j} s_k^{j,i}. \tag{2}$$

Initially, as shown on Line 1, $s_1^{i,j} = s_1^{j,i}$ is the 0 vector (i.e., all elements 0) for all $i \in \mathcal{N}_j$. Thus, the perturbation d_1^j computed in iteration 1 is also the 0 vector. As shown on Line 6 of Algorithm 1, in iteration $k \geq 1$, agent j sends to each neighbor i a random vector $s_{k+1}^{j,i}$ and then (on Line 7) it receives random vector $s_{k+1}^{i,j}$ from each neighbor i. These random vectors are then used to compute the random perturbations in Line 4 of the next iteration. Due to the manner in which d_j^k is computed, we obtain the following invariant for all iterations $k \geq 1$.

$$\sum_{j \in \mathcal{V}} d_k^j \ = \ 0.$$

In our analysis, we will assume that $\|s_k^{j,i}\| \leq \Delta/(2n)$, for all i, j, k, where constant Δ is a parameter in our analysis, and $n = |\mathcal{V}|$ is the number of agents. Computation of perturbation d_k^j, as shown in (2), then implies that $\|d_k^j\| \leq \Delta$. As elaborated later, there is a trade-off between privacy and convergence rate of the algorithm, with larger Δ resulting in slower convergence rate [3].

C. RSS-LB *Algorithm*

Our second algorithm is called *Randomized State Sharing−Locally Balanced* algorithm (RSS-LB). Recall that in RSS-NB Algorithm 1, each agent shares an identical perturbed estimate with its neighbors. Instead, in RSS-LB, each

[2] B_k has the property that entries $B_k[i,j]$ and $B_k[j,i]$ are non-zero if and only if $i \in \mathcal{N}_j$. Recall that the underlying network is assumed to consist of bidirectional links. Therefore, $i \in \mathcal{N}_j$ implies $j \in \mathcal{N}_i$.

[3] Agents need not know Δ. As long as $\|s_k^{j,i}\|$ has finite bound, we get bounded $\|d_k^j\|$ and correctness results hold with $\Delta \triangleq \max_j \|d_k^j\|$.

Algorithm 1 RSS-NB Algorithm

1: Input: α_k $(k \geq 1)$, MAX_ITER.
Initialization: $x_0^j \in \mathcal{X},\ \forall j \in \mathcal{V}$ and
$s_1^{i,j} = s_1^{j,i} = 0,\ \forall j \in \mathcal{V}, i \in \mathcal{N}_j$.
2: The steps performed by each agent $j \in \mathcal{V}$:
3: **for** k = 1 to MAX_ITER **do**
4: Compute perturbation d_k^j:
$d_k^j = \sum_{i \in \mathcal{N}_j} s_k^{i,j} - \sum_{i \in \mathcal{N}_j} s_k^{j,i}$
5: Compute perturbed state w_k^j:
$w_k^j = x_k^j + \alpha_k d_k^j$
6: Send w_k^j and a random vector $s_{k+1}^{j,i}$ to $i \in \mathcal{N}_j$.
7: Receive w_k^i and $s_{k+1}^{i,j}$ from $i \in \mathcal{N}_j$.
8: Information Fusion:
$v_k^j = \sum_{i \in \mathcal{N}_j} B_k[j,i] w_k^i$
9: Projected Gradient Descent:
$x_{k+1}^j = \mathcal{P}_{\mathcal{X}} \left[v_k^j - \alpha_k \nabla f_j(v_k^j) \right]$
10: **end for**

Algorithm 2 RSS-LB Algorithm

1: Input: α_k $(k \geq 1)$, MAX_ITER.
Initialization: $x_0^j \in \mathcal{X},\ \forall j \in \mathcal{V}$.
2: The steps performed by each agent $j \in \mathcal{V}$:
3: **for** k = 1 to MAX_ITER **do**
4: Choose random vector $d_k^{j,i}$, $i \in \mathcal{N}_j$, such that,
$\sum_{i \in \mathcal{N}_j} B_k[i,j] d_k^{j,i} = 0$.
5: Compute perturbed state $w_k^{j,i}$:
$w_k^{j,i} = x_k^j + \alpha_k d_k^{j,i}$
6: Send $w_k^{j,i}$ to each $i \in \mathcal{N}_j$.
7: Receive $w_k^{i,j}$ from each $i \in \mathcal{N}_j$.
8: Information Fusion:
$v_k^j = \sum_{i \in \mathcal{N}_j} B_k[j,i] w_k^{i,j}$
9: Projected Gradient Descent:
$x_{k+1}^j = \mathcal{P}_{\mathcal{X}} \left[v_k^j - \alpha_k \nabla f_j(v_k^j) \right]$
10: **end for**

agent shares potentially distinct perturbed state estimates with different neighbors. The pseudo-code for RSS-LB is presented as Algorithm 2 below.

On Line 4 of Algorithm 2, in iteration k, agent j chooses a noise vector $d_k^{j,i} \in \mathbb{R}^D$ for each $i \in \mathcal{N}_j$ such that $d_k^{j,j} = 0$, $\|d_k^{j,i}\| \leq \Delta$, where constant Δ is a parameter of the algorithm, and

$$\sum_{i \in \mathcal{N}_j} B_k[i,j] d_k^{j,i} = 0. \tag{3}$$

Here, matrix B_k is identical to that used in the information fusion step in Line 8. Observe that each agent j uses $B_k[j,i]$, $i \in \mathcal{N}_j$, in the information fusion step, and $B_k[i,j]$, $i \in \mathcal{N}_j$, in the computation of above noise vectors. In both cases, the matrix elements used by agent j correspond only to its neighbors in the network. Since the random vectors generated by each agent j are *locally balanced*, as per (3) above, the agents do not need to cooperate in generating the perturbations (unlike the RSS-NB algorithm).

Using $d_k^{j,i}$ as the perturbation for neighbor i, in Line 5 of Algorithm 2, agent j computes the perturbed state estimate $w_k^{j,i}$ to be sent to neighbor i, as follows.

$$w_k^{j,i} = x_k^j + \alpha_k d_k^{j,i}. \tag{4}$$

α_k here is the step size, which is also used in the information fusion step in Line 8. Properties satisfied by α_k are identical to thos in the DGD Algorithm 0.

Next, in Lines 6 and 7 of Algorithm 2, agent j sends $w_k^{j,i}$ to each neighbor i and receives perturbed estimate $w_k^{i,j}$ from each neighbor i. Agent j performs the information fusion step in Line 8 followed by projected gradient descent in Line 9, similar to the previous algorithms.

D. FS *Algorithm*

The *function sharing* algorithm FS presented in this section can be viewed as a special case of the RSS-NB algorithm. In this special case of RSS-NB, the random vector $s_k^{j,i}$ computed by agent j is a function of its state estimate x_k^j, where the function is independent of k. Thus, the function sharing algorithm uses *state-dependent* random vectors. The pseudo-code for function sharing is presented in Algorithm 3 below using random functions, instead of state-dependent random vectors. However, the behavior of Algorithm 3 is equivalent to using state-dependent noise in RSS-NB.

In Line 1 of Algorithm 3, each agent j selects a function $s^{j,i}(x)$ to be sent to neighbor i in Line 2. These functions are exchanged by the agents. Agent j then uses them in Line 3 to compute the noise function, which is, in turn, used to compute an obfuscated local objective function $\widehat{f}_j(x)$. Finally, the agents perform DGD Algorithm 0 with each agent j using $\widehat{f}_j(x)$ as its objective function. We assume that $s^{j,i}(x)$ have bounded and Lipschitz gradients. This implies the obfuscated functions $\widehat{f}_j(x)$ satisfy Assumption 2. The obfuscated objective function $\widehat{f}_j(x)$ is not necessarily convex. Despite this, the correctness of this algorithm can be proved using the following observations:

$$\sum_{j \in \mathcal{V}} p_j(x) = 0, \text{ and} \tag{5}$$

$$\sum_{j \in \mathcal{V}} \widehat{f}_j(x) = \sum_{j \in \mathcal{V}} f_j(x) \ = \ f(x) \tag{6}$$

Effectively, Algorithm 3 minimizes a *convex sum of non-convex functions*. Distributed optimization of a convex sum of non-convex functions, albeit with an additional assumption of *strong convexity* of $f(x)$, was also addressed in [35], wherein the correctness is shown using Lyapunov stability arguments. However, [35] also does not address how privacy may be achieved. Additionally, our approach for improving privacy is more general than function sharing, as exemplified by algorithms RSS-NB and RSS-LB.

Algorithm 3 Function Sharing (FS) Algorithm

1: The steps performed by each agent $j \in \mathcal{V}$
2: Select a function $s^{j,i}(x)$, $\forall i \in \mathcal{N}_j$.
3: Agent j sends function $s^{j,i}(x)$ to each $i \in \mathcal{N}_j$.
4: Agent j computes a noise function $p_j(x)$ and then the obfuscated local objective function $\widehat{f}_j(x)$ as follows:

$$p_j(x) = \sum_{i \in \mathcal{N}_j} s^{i,j}(x) - \sum_{i \in \mathcal{N}_j} s^{j,i}(x). \tag{7}$$

$$\widehat{f}_j(x) \triangleq f_j(x) + p_j(x) \tag{8}$$

5: Perform DGD (Algorithm 0) wherein agent j uses $\widehat{f}_j(x)$ as its local objective function instead of $f_j(x)$.

IV. Main Results

Problem 1 in Section II identifies the requirement for correctness of the proposed algorithms.

Theorem 1. *Under Assumptions 1 and 2,* RSS-NB *Algorithm 1,* RSS-LB *Algorithm 2 and* FS *Algorithm 3 solve distributed optimization Problem 1 asymptotically.*

Theorem 1 implies that the sequence of iterates $\{x_k^j\}$, generated by each agent j converges to an optimum in $\mathcal{X}^*$ asymptotically (as MAX_ITER $\rightarrow \infty$), despite the introduction of perturbations.

Now we discuss privacy improvement achieved by our algorithms. We consider an adversary that compromises a set of up to τ agents, denoted as $\mathcal{A}$ (thus, $|\mathcal{A}| \leq \tau$). The adversary can observe everything that each agent in $\mathcal{A}$ observes. In particular, the adversary has the knowledge of the local objective functions of agents in $\mathcal{A}$, their state, and their communication to and from all their neighbors. Furthermore, the adversary knows the network topology.

The goal here is to prevent the adversary from learning the local objective function of any agent $i \notin \mathcal{A}$. The introduction of perturbations in the state estimates helps improve privacy, by creating an ambiguity in the following sense. To be able to exactly determine $f_i(x)$ for any $i \notin \mathcal{A}$, the adversary's observations of the communication to and from agents in $\mathcal{A}$ has to be compatible with the actual $f_i(x)$, but not with any other possible choice for the local objective function of agent i. The larger the set of possible local objective functions of agent i that are compatible with the adversary's observations, greater is the ambiguity. The introduction of noise naturally increases this ambiguity, with higher Δ (noise parameter) resulting in greater privacy. However, this improved privacy comes with a performance cost, as Theorem 3 will show. Before we discuss Theorem 3, we first present more precise claims for privacy for the FS algorithm.

Privacy Claims: Let $\mathcal{F}$ denote the set of all instances of Problem 1. Each element of $\mathcal{F}$, say $\{g_1(x), g_2(x), \cdots, g_n(x)\}$ corresponds to an instance of Problem 1, where the $g_i(x)$ become the local objective functions for each agent i (satisfying Assumptions 1, 2). Thus any element of $\mathcal{F}$ can represent an instance of distributed optimization problem similar to Problem 1. We consider the scenario when local objective functions are polynomials of bounded degree. We call any polynomial with bounded degree satisfying convexity and gradient Lipschitzness as feasible local objective function. Theorem 2 makes a claim regarding the privacy achieved using *function sharing* under this scenario.

Definition 1. *Recall that $\mathcal{F}$ is the set of all possible instances of Problem 1. The adversary's observations are said to be compatible with problem instance $\{g_1(x), g_2(x), \cdots, g_n(x)\} \in \mathcal{F}$ if the information available to the adversary may be produced when agent i's local objective function is $g_i(x)$ for each $i \in \mathcal{V}$.*

Theorem 2. *Let the local objective function of each agent be restricted to be a polynomial of a bounded degree, and let Assumptions 1 and 2 hold. Consider an execution of the* FS *algorithm in which the local objective function of each agent i is $f_i(x)$. Then* FS *algorithm provides the following privacy guarantees:*

(P1) *Let the network graph $\mathcal{G}$ have a minimum degree $\geq \tau + 1$. For any agent $i \notin \mathcal{A}$, choose any feasible local objective function $g_i(x) \neq f_i(x)$. The adversary's observations in the above execution are compatible with at least one feasible problem in $\mathcal{F}$ in which agent i's local objective function equals $g_i(x)$. In other words, the adversary cannot learn function $f_i(x)$ for $i \notin \mathcal{A}$.*

(P2) *Let the network graph $\mathcal{G}$ have vertex connectivity $\geq \tau + 1$. For each $\mathcal{I} \subset \mathcal{V} - \mathcal{A}$, choose a feasible local objective function $g_i(x) \neq f_i(x)$ for each $i \in \mathcal{I}$. The adversary's observations in the above execution are compatible at least one feasible problem in $\mathcal{F}$ wherein, for $i \in \mathcal{I}$, agent i's local objective function is $g_i(x)$. In other words, the adversary cannot learn $\sum_{i \in \mathcal{I}} f_i(x)$.*

The proof for property (P2) in Theorem 2 is sketched in Section VI. Property (P1) can be proved similarly.

Convergence-Privacy Trade-off: Addition of perturbations to the state estimates can improve privacy, however, it also degrades the convergence rate. Analogous to the finite-time analysis presented in [11], the theorem below assumes $\alpha_k = 1/\sqrt{k}$, and provides a convergence result for a weighted time-average of the state estimates $\widehat{x}_T^j$ defined below.

Theorem 3. *Let estimates $\{x_k^j\}$ be generated by* RSS-NB *or* RSS-LB *with $\alpha_k = 1/\sqrt{k}$. For each $j \in \mathcal{V}$, let*

$$\widehat{x}_T^j = \frac{\sum_{k=1}^T \alpha_k x_k^j}{\sum_{k=1}^T \alpha_k}.$$

Then,

$$f(\widehat{x}_T^j) - f(x^*) \leq \mathcal{O}\left((1+\Delta^2)\frac{\log(T)}{\sqrt{T}}\right).$$

Section V presents the proof sketch. The above theorem shows that the gap between the optimal function value and function value at the time-average of state estimates ($\widehat{x}_k^j$) has

a gap that is quadratic in noise bound Δ. The dependence on time T in the convergence result above is similar to that for DGD in [11], and is a consequence of the consensus-based local gradient method used here. The quadratic dependence on Δ is a consequence of structured randomization. Larger Δ results in slower convergence, however, would result in larger randomness in the iterates, improving privacy.

Random perturbations used in algorithms RSS-NB and RSS-LB cause a slowdown in convergence, however, do not introduce an error in the outcome. This is different from $\tilde{\epsilon}$-Differential Privacy where perturbations result in slowdown in addition to an error of the order of $\mathcal{O}(1/\tilde{\epsilon}^2)$ [29].

V. Performance Analysis

We sketch the analysis of RSS-NB here. Analysis of RSS-LB and FS follows similar structure. For brevity, only key results are presented here. Detailed proofs are available in [36]–[38]. We often refer to the *state estimate* of an agent as its *iterate*. Define iterate average ($\bar{x}_k$, at iteration k) and the disagreement of iterate x_k^j agent j with $\bar{x}_k$ as,

$$\bar{x}_k = \frac{1}{n}\sum_{j=1}^{n} x_k^j, \text{ and } \delta_k^j = x_k^j - \bar{x}_k.$$

The computation on Line 7 of RSS-NB Algorithm 1 can be represented using "true state", denoted as $\widehat{v}_k^j$, and a perturbation, e_k^j, as follows.

$$\begin{aligned} \widehat{v}_k^j &= \sum_{i\in\mathcal{N}_j} B_k[j,i]x_k^i,\ e_k^j = \sum_{i\in\mathcal{N}_j} B_k[j,i]d_k^j \text{ and} \quad (9)\\ v_k^j &= \sum_{i\in\mathcal{N}_j} B_k[j,i]w_k^i \ = \ \widehat{v}_k^j + \alpha_k e_k^j \end{aligned}$$

Since $\sum_j d_k^j = 0$ and B_k is doubly stochastic, we get, $\sum_j e_k^j = 0$.

Now we can represent the projected gradient descent step (Line 8 of Algorithm 1) as,

$$x_{k+1}^j = \mathcal{P}_{\mathcal{X}}\left[\widehat{v}_k^j - \alpha_k\left(\nabla f_j(v_k^j) - e_k^j\right)\right]. \quad (10)$$

In the above expression, the perturbation can be viewed simply as noise in the gradient. This perspective is useful for the analysis. Using a result from [10] on linear convergence of product of doubly stochastic matrices, we obtain a bound on disagreement $\|\delta_k^j\|$ in Lemma 1 below.

Lemma 1. *For constant $\beta < 1$ and constant θ that both only depend on the network $\mathcal{G}$ and the stochastic matrices B_k,*

$$\max_{j\in\mathcal{V}}\|\delta_{k+1}^j\| \le n\theta\beta^{k+1}\max_{i\in\mathcal{V}}\|x_0^i\| + 2\alpha_k(L+\Delta) + n\theta(L+\Delta)\sum_{l=1}^{k}\beta^{k+1-l}\alpha_{l-1}$$

Proof. Detailed proof is included in [36]. □

Lemma 1 can be used to show that the iterates maintained by the different agents asymptotically reach consensus. Lemma 2 below provides a bound on the distance between iterates and the optimum.

Lemma 2. *The following holds for all $y \in \mathcal{X}$.*

$\eta_{k+1}^2 \le (1+F_k)\,\eta_k^2 - 2\alpha_k\,(f(\bar{x}_k) - f(y)) + H_k$, *where*

$\eta_k^2 = \sum_{j=1}^{n}\|x_k^j - y\|^2$, $F_k = \alpha_k N\left(\max_{j\in\mathcal{V}}\|\delta_k^j\| + \alpha_k\Delta\right)$, *and*

$H_k = \alpha_k n\left(2\max_{j\in\mathcal{V}}\|\delta_k^j\|(L + \frac{N}{2} + \Delta) + \alpha_k\left(N\Delta + (L+\Delta)^2\right)\right]$.

The expressions in Lemma 2 has the same structure as supermartingale convergence result from [39]. We can show that $\sum_k H_k < \infty$ and $\sum_k F_k < \infty$. Then using the result from [39] asymptotic convergence of the iterate average $\bar{x}_k$ to an optimum $x^* \in \mathcal{X}^*$ can be proved, proving Theorem 1.

Proof of Theorem 3: Next, we sketch the proof of Theorem 3, which uses Lemma 2. As discussed earlier, Theorem 3 assumes $\alpha_k = 1/\sqrt{k}$. Recall the definition of $\widehat{x}_T^j$ in Theorem 3. Let $\widehat{\bar{x}}_T = \frac{1}{n}\sum_{j=1}^{n}\widehat{x}_T^j$. Observing that $\widehat{\bar{x}}_T$ also equals $\sum_{k=1}^{T}\alpha_k\bar{x}_k$ and using the fact that $f(x)$ is convex,

$$f(\widehat{\bar{x}}_T) - f^* \le \frac{\sum_{k=1}^{T}\alpha_k f(\bar{x}_k)}{\sum_{k=1}^{T}\alpha_k} - f^* = \frac{\sum_{k=1}^{T}\alpha_k\,(f(\bar{x}_k) - f^*)}{\sum_{k=1}^{T}\alpha_k}$$

Lemma 2 and the observation $\sum_{k=1}^{T}\alpha_k \ge \sqrt{T}$ yields:

$$\begin{aligned} f(\widehat{\bar{x}}_T) - f^* &\le \frac{\sum_{k=1}^{T}\left((1+F_k)\eta_k^2 - \eta_{k+1}^2 + H_k\right)}{\sum_{k=1}^{T}\alpha_k}\\ &\le \frac{\eta_1^2 + \sum_{k=1}^{T}\left(F_k\eta_k^2 + H_k\right)}{\sqrt{T}} \end{aligned}$$

Next we bound $\sum_{k=1}^{T}F_k$ and $\sum_{k=1}^{T}H_k$.

$$\begin{aligned} \sum_{k=1}^{T}F_k &\le 2N\sum_{k=1}^{T}\alpha_k\max_j\|\delta_k^j\| + 2N\Delta(\log(T)+1)\\ \sum_{k=1}^{T}H_k &\le 2n(L+\frac{N}{2}+\Delta)\sum_{k=1}^{T}\alpha_k\max_j\|\delta_k^j\|\\ &\quad + n[(L+\Delta)^2 + N\Delta](\log(T)+1) \end{aligned}$$

Use Lemma 1 to bound $\sum_{k=1}^{T}\alpha_k\max_j\|\delta_k^j\|$.

$$f(\widehat{\bar{x}}_T) - f^* \le \frac{C_0 + C_1\log(T) + C_2\log(T-1)}{\sqrt{T}}$$

We then use the Lipschitzness of $f(x)$ to arrive at,

$$\begin{aligned} f(\widehat{x}_T^j) - f^* &= f(\widehat{x}_T^j) - f(\widehat{\bar{x}}_T) + f(\widehat{\bar{x}}_T) - f^*\\ &\le L\|\widehat{x}_T^j - \widehat{\bar{x}}_T\| + \frac{C_0 + (C_1+C_2)\log(T)}{\sqrt{T}}\\ &\quad \text{where } C_1, C_2 = \mathcal{O}(\Delta^2)\\ &\le \mathcal{O}\left((1+\Delta^2)\frac{\log(T)}{\sqrt{T}}\right) \end{aligned}$$

□

VI. Privacy with Function Sharing

In this section we consider a special case of Problem 1. Assume that all objective functions $f_i(x)$ are polynomials with degree $\le d$. Consequently, $f(x)$ is a polynomial with $\deg(f(x)) \le d$. Polynomial objective functions often appear in robotics applications such as rendezvous and formation

flying. We now prove property (P2) in Theorem 2; the proof of property (P1) can be obtained similarly.

Proof of Theorem 2 (P2): In particular, we use a constructive approach to show that given any execution, any feasible candidate for local objective functions of nodes in any subset $\mathcal{I} \subset \mathcal{V} - \mathcal{A}$ is compatible with the adversary's observations.

Suppose that the actual local objective function of each agent i in a given execution is $f_i(x)$. Consider subset $\mathcal{I} \subset \mathcal{V} - \mathcal{A}$, and consider any feasible local objective function $g_i(x)$ for each $i \in \mathcal{I}$. Now, for any $i \in \mathcal{A}$, let $g_i(x) = f_i(x)$ (adversary observes own objective functions). Also, since the functions are polynomials of bounded degree, it should be easy to see that, for each $i \in \mathcal{V} - \mathcal{A} - \mathcal{I}$, we can find local objective functions $g_i(x)$ such that $\sum_{i \in \mathcal{V}-\mathcal{A}} g_i(x) = \sum_{i \in \mathcal{V}-\mathcal{A}} f_i(x)$, for all $x \in \mathcal{X}$. Thus, for the given functions $g_i(x)$ for $i \in \mathcal{I}$, we have found feasible local objective functions $g_i(x)$ for all agents such that – i) the local objective functions of compromised agents are identical to those in the actual execution, and ii) the sum of objective functions of "good" nodes is preserved $\sum_{i \in \mathcal{V}-\mathcal{A}} g_i(x) = \sum_{i \in \mathcal{V}-\mathcal{A}} f_i(x)$.

Recall that the *function sharing* algorithm adds noise functions to obtain perturbed function $\widehat{f}_i(x)$ at each agent $i \in \mathcal{V}$. In particular, agent j sends to each neighboring agent i a noise function, say $s^{j,i}(x)$, and subsequently computes $\widehat{f}_i(x)$ using the noise functions it sent to neighbors and the noise functions received from the neighbors.

When the vertex connectivity of the graph is at least $\tau + 1$ it is easy to show that, for local objective functions $\{g_1(x), g_2(x), \cdots, g_n(x)\}$ defined above, each agent $j \in \mathcal{V} - \mathcal{A}$ can select noise functions, say $t^{j,i}(x)$ for each neighbor i, with the following properties:

- For each $j \in \mathcal{V} - \mathcal{A}$ and neighbor i of j such that $i \in \mathcal{A}$, $t^{j,i}(x) = s^{j,i}(x)$ for all $x \in \mathcal{X}$. That is, the noise functions exchanged with agents in $\mathcal{A}$ are unchanged.
- For each $j \in \mathcal{V} - \mathcal{A}$,

$$g_j(x) + \sum_{i \in \mathcal{N}_j} t^{i,j}(x) - \sum_{i \in \mathcal{N}_j} t^{j,i}(x) = \widehat{f}_j(x).$$

  That is, the obfuscated function of each agent in $\mathcal{V} - \mathcal{A}$ remains the same as that in the original execution.

Due to the above two properties, the observations of the adversary in the above execution will be identical to those in the original execution. Thus, the adversary cannot distinguish between the two executions. This, in turn, implies property (P2) in Theorem 2. Property (P1) can be proved similarly. □

VII. Experimental Results

We now provide some experimental results for RSS-NB and RSS-LB algorithms. We present two sets of experiments. First, we show that RSS-NB and RSS-LB correctly solve distributed optimization of polynomial objective functions. Next, we apply our algorithms in the context of machine learning for handwritten digit classification (MNIST dataset) and document classification (Reuters dataset).

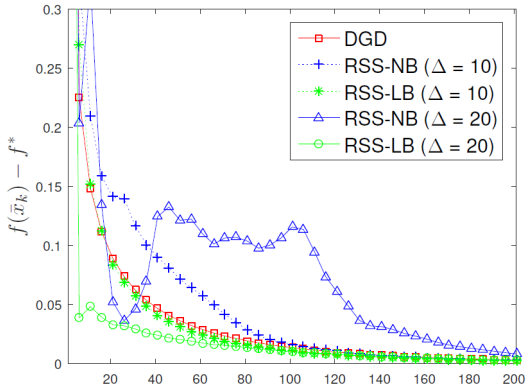

Fig. 1: Function suboptimality v/s iterations.

Polynomial Optimization: We solve polynomial optimization on a network of 5 agents that form a cycle. The objective functions of the 5 agents are chosen as $f_1(x) = x^2$, $f_2(x) = x^4$, $f_3(x) = x^2 + x^4$, $f_4(x) = x^2 + 0.5x^4$, and $f_5(x) = 0.5x^2 + x^4$. The aggregate function $f(x) = 2.5(x^2 + x^4)$. We consider $\mathcal{X} = [-30, 30]$ and have the same initialization for all cases. Simulation results in Figure 1 show that the two algorithms converge to the optimum $x^* = 0$. Large Δ results in larger perturbations and the convergence is slower. For smaller Δ, as expected, the performance of both RSS-NB and RSS-LB is closer to DGD.

Machine Learning: We consider two classification problems. We use a deep neural network [7] for digit recognition using the MNIST dataset [40] and regularized logistic regression for the Reuters dataset [41]. We use two graph topologies: a cycle of 5 agents (namely, C_5) and a complete graph of 5 agents (namely, K_5). Due to the high cost of computing gradients, we use stochastic gradients computed on minibatches of local data instead of full gradients. We perform consensus only every 40 gradient descent steps, decreasing communication overhead, while still retaining the

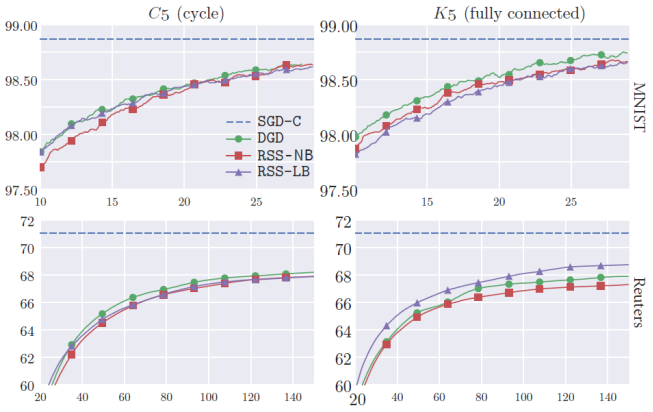

Fig. 2: Accuracy for MNIST (top row) and Reuters (bottom row) with C_5 (left) and K_5 (right) topologies. Y-axis is the testing accuracy in % and X-axis is epochs. SGD-C is a centralized solution.

performance (accuracy). Figure 2 shows convergence results for DGD, RSS-NB and RSS-LB, and a centralized algorithm SGD-C, which demonstrate that we achieve testing accuracy comparable to a centralized solution SGD-C. RSS-NB and RSS-LB perform comparable to DGD for both datasets.

Note that our algorithms converge quickly despite deep learning problem being non-convex and use of stochastic gradients. This underlines the fact that although our analysis is for convex problems, our algorithm behaves well with deep-learning problem which is highly non-convex.

VIII. Conclusion

In this paper, we develop and analyze iterative distributed optimization algorithms RSS-NB, RSS-LB and FS that exploit *structured randomness* to improve privacy, while maintaining accuracy. We prove convergence and develop trade-off between convergence rate and the bound on perturbation. We provide privacy claims and proofs for the FS algorithm, which is a special case of RSS-NB. We apply versions of RSS-NB and RSS-LB to distributed machine learning, and evaluating their effectiveness for digit recognition (MNIST) and document classification (Reuters dataset).

References

[1] 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.

[2] 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.

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

[4] 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.

[5] 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.

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

[7] 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.

[8] F. Pasqualetti, F. Dörfler, and F. Bullo, "Cyber-physical security via geometric control: Distributed monitoring and malicious attacks," in *51st IEEE Conference on Decision and Control (CDC)*, pp. 3418–3425, IEEE, 2012.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.

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

[20] 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.

[21] E. Wei and A. Ozdaglar, "On the O(1/k) convergence of asynchronous distributed alternating direction method of multipliers," in *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, pp. 551–554, IEEE, 2013.

[22] A. Agarwal and J. C. Duchi, "Distributed delayed stochastic optimization," in *NIPS*, pp. 873–881, 2011.

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

[24] B. Pinkas, "Cryptographic techniques for privacy-preserving data mining," *ACM Sigkdd Expl. Newsletter*, vol. 4, no. 2, pp. 12–19, 2002.

[25] Y. Hong, J. Vaidya, N. Rizzo, and Q. Liu, "Privacy preserving linear programming," *arXiv preprint arXiv:1610.02339*, 2016.

[26] Z. Huang, S. Mitra, and N. Vaidya, "Differentially private distributed optimization," in *Proceedings of the 2015 International Conference on Distributed Computing and Networking*, p. 4, ACM, 2015.

[27] E. Nozari, P. Tallapragada, and J. Cortés, "Differentially private distributed convex optimization via functional perturbation," *arXiv preprint arXiv:1512.00369*, 2015.

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

[33] L. Xiao, S. Boyd, and S. Lall, "Distributed average consensus with time-varying metropolis weights," *Automatica*, 2006.

[34] 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.

[35] K. Kvaternik and L. Pavel, "Lyapunov analysis of a distributed optimization scheme," in *NetGCooP-2011*, pp. 1–5, IEEE, 2011.

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

[37] S. Gade and N. H. Vaidya, "Distributed optimization of convex sum of non-convex functions," *arXiv preprint arXiv:1608.05401*, 2016.

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

[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] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[41] O. Ludwig, "Deep learning with eigenvalue decay regularizer," *CoRR*, vol. abs/1604.06985, 2016.