Privacy-Preserving Distributed Multi-Task Learning with Asynchronous Updates

Liyang Xie
Michigan State University
xieliyan@msu.edu

Inci M. Baytas
Michigan State University
baytasin@msu.edu

Kaixiang Lin
Michigan State University
linkaixi@msu.edu

Jiayu Zhou
Michigan State University
jiayuz@msu.edu

ABSTRACT

Many data mining applications involve a set of related learning tasks. Multi-task learning (MTL) is a learning paradigm that improves generalization performance by transferring knowledge among those tasks. MTL has attracted so much attention in the community, and various algorithms have been successfully developed. Recently, distributed MTL has also been studied for related tasks whose data is distributed across different geographical regions. One prominent challenge of the distributed MTL frameworks is to maintain the privacy of the data. The distributed data may contain sensitive and private information such as patients' records and registers of a company. In such cases, distributed MTL frameworks are required to preserve the privacy of the data. In this paper, we propose a novel privacy-preserving distributed MTL framework to address this challenge. A privacy-preserving proximal gradient algorithm, which asynchronously updates models of the learning tasks, is introduced to solve a general class of MTL formulations. The proposed asynchronous approach is robust against network delays and provides a guaranteed differential privacy through carefully designed perturbation. Theoretical guarantees of the proposed algorithm are derived and supported by the extensive experimental results.

CCS CONCEPTS

•**Computing methodologies** →**Multi-task learning;** *Distributed algorithms;* •**Security and privacy** →**Privacy protections;**

KEYWORDS

Multi-task Learning; Differential Privacy; Asynchronous Proximal Optimization

1 INTRODUCTION

With the onset of the big data era, designing efficient algorithms to leverage massive amount of available data has become a prominent research goal. A huge sample size is one of the desired aspects of a learning problem since it helps to improve the generalization performance. However, it is not always possible to use all of the available data within a single learning task framework. For instance, data might come from different distributions which result in multiple learning tasks. These multiple tasks are inherently related since they generally share the same target. Multi-task learning (MTL) is a powerful technique to learn the multiple related tasks simultaneously to improve the generalization performance compared to learning several tasks individually. MTL performs inductive knowledge transfer among the related tasks through learning a shared representation. Several types of shared representations can be encountered in MTL literature such as shared feature subspace [2, 11], shared feature subset [12, 31, 46, 50], clustering structure [47] and shared parameters in neural networks [8]. One of the popular approaches to achieve shared representation is structured regularization [21, 48].

MTL is especially useful when the amount and the quality of the data for each task are not sufficient to learn a powerful model. Even in the big data era, we may still have inadequate amount of data to support one specific learning task. For example, a hospital may collect data from tens of thousands of patients to build predictive models to study certain diseases. While the sample size is large, the distribution of the patients may be skewed because of the demographics of the hospital location. Therefore, the big data does not necessarily assure "enough data" due to many factors such as high dimensional feature space, high model complexity, and skewed distribution. For this reason, MTL remains as an important research area in big data analysis.

There are two major challenges when applying MTL to big data. Firstly, data of each learning task may be stored in different servers across different geographical regions. It is not always feasible to aggregate the distributed data into a central location due to bandwidth and security restrictions. In literature, several distributed MTL approaches have been developed to tackle the aforementioned challenge [34, 35]. As most of the MTL algorithms depend on optimization with iterative procedures, such as gradient descent or proximal gradient, traditional synchronized optimization algorithms may lead to prohibitive training time. Another major challenge is that how we can protect the privacy of the sensitive data in distributed MTL frameworks. One scenario, where privacy protection is necessary, is medical studies conducted in different hospitals. Hospitals in different geographical regions may make a joint effort to study diseases (e.g., the ENIGMA project funded by National Institutes of Health [42]), while the raw patient data cannot

DOI: http://dx.doi.org/10.1145/3097983.3098152

be shared because of the sensitive nature and/or the privacy policy enforced. Even though hospitals may learn individual models by distributed MTL and by only transferring gradient information instead of the raw data, untrusted observers in the network may still violate the privacy by manipulating the protocol. So far there have been few prior studies on privacy preserving algorithms for distributed asynchronous MTL.

In this paper, we propose a novel privacy-preserving distributed MTL framework to address the aforementioned challenges. A privacy-preserving proximal gradient algorithm, which asynchronously updates models of the learning tasks, is introduced to solve a general class of MTL formulations. The proposed asynchronous approach is robust against network delays and provides a guaranteed differential privacy through carefully designed perturbation. We conducted extensive experiments to demonstrate the effectiveness and the accuracy of the proposed algorithm and its theoretical properties. The key contributions of this paper are as follows: 1) We propose a novel differentially private proximal gradient algorithm combined with distributed asynchronous MTL framework to handle the aforementioned challenges. 2) We derive theoretical proofs of the privacy guarantees of the proposed algorithm under two learning problems. 3) We provide extensive empirical experiments to evaluate our algorithm and its theory. To our knowledge, this is the first differentially private proximal gradient algorithm used in the asynchronous distributed MTL framework. Our method can be applied in various scenarios and can be easily generalized.

The rest of the paper is organized as follows: In Section 2, we review related literature. In Section 3, we discuss the proposed algorithm and the proofs of the privacy guarantees. In Section 4, we present our experimental results of the empirical studies supporting the theoretical properties of our algorithm. Finally in Section 5, we conclude our work.

2 RELATED WORK

In this section, we provide a literature review of distributed multi-task learning (MTL), differential privacy and asynchronous proximal optimization.

Distributed Multi-Task Learning. Our study is motivated by the learning problems where private and sensitive data is distributed across different local tasks, and tasks are related to each other. In such cases, centralized MTL methods face two challenges: data privacy and bandwidth restrictions. On the other hand, distributed MTL frameworks provide an efficient solution to deal with the aforementioned challenges. In a distributed MTL framework, only task model or its gradient is transferred between task node and central server, hence it significantly reduces the communication time. Furthermore, there is no data point involved in the communication. Therefore the data privacy is guaranteed in several aspects[1]. For example, Dinuzzo *et al.* [15] proposed a client-server architecture to simultaneously solve the multiple learning tasks from distributed databases based on regularization theory and kernel methods. Each client performs an individual learning task and the server collects the data and performs the information fusion. Thus, each client can exploit the informative content of all the data coming from different tasks without any access to the private data of other clients.

Mateos-Núñez *et al.* [35] explored a separable convex optimization problem and a separable saddle-point reformulation which can be extended to distributed MTL framework. The computation of the gradient is divided into local agents. Another study [29] proposed distributed online MTL (DOM) framework, which includes collaborative local learning in local task nodes and asynchronous online MTL method as the global learning in central node. Recently, Liu *et al.* [34] proposed a distributed MTL framework that can simultaneously learn task relationships and task-specific predictive models from geo-distributed task data. In [4], Baytas *et al.* proposed an asynchronous MTL framework where the server does not wait for the task nodes to complete their computations before the proximal update of the model is performed. They used backward-forward splitting algorithm to perform gradient step in the task nodes and the proximal mapping of the model matrix in the server node under a regularized distributed MTL setting. Our method is very different from [4]. Instead of updating whole model in each task, we decompose the model into an independent part and a shared part, which are separately updated in the server and the task nodes. This saves communication resources compared with prevous works and is also a novel MTL framework. We also protect the privacy of the independent part in a differentially private sense to prevent any adversary at the server side. This is achieved by adding noise on two levels, which is a new differential privacy mechanism.

Differential Privacy. Differentially private learning has been widely studied in the literature, examples including Chaudhuri *et al.* [9, 10] where an efficient differentially private empirical risk minimization was developed, Jain and Thakurta [26] studied kernel learning under three practical models, and Kusner *et al.* [32] focused on Bayesian learning with near-optimal performance. Thakurta and Smith [41] studied feature selection with stability property, whereas Xiao *et al.* [45] utilized wavelet transform which can be applied on both ordinal and nominal data. Friedman and Schuster [22] used a data mining approach that demonstrates the trade off between privacy, accuracy, and the sample size, and Li *et al.* [33] optimized linear counting queries. Shokri and Shmatikov [40] described an asynchronous update scheme to train deep models in a distributed and differentially private manner. Rajkumar and Agarwal [38] used stochastic gradient descent which takes long term effects into consideration, and they boosted the methods in Dwork's work [20]. Differentially private MTL was studied in Gupta *et al.* [25], where the authors proposed a differentially private algorithm using noisy task relation matrix. Also they developed an attribute-wise noise addition scheme that significantly improves the utility of their proposed method.

Asynchronous Proximal Optimization. In [37], an asynchronous parallel coordinate descent framework was proposed for a general class of optimization problems. They formulated several optimization problems as finding fixed point of a non-expansive operator. However, the proposed method in [37] did not focus on proximal alternating linearized minimization (PALM). PALM is a powerful block coordinate descent method that is widely used in machine learning problems dealing with non-convex and non-smooth functions. Bolte *et al.* [7] proposed the aforementioned

[1]Both the privacy in common sense and the differential privacy are the concepts used in this paper.

idea and derived a new self-contained convergence analysis framework. The framework leverages Kurdyka-Lojasiewicz property and guarantees convergence to a composite objective. In [13, 14], new extensions of PALM, where asynchronous update of variable blocks is allowed, were developed. Richtárik and Takáč [39] also used randomized coordinate descent methods to solve minimization problem with large scale data, and provided a bound that depends on the data and partition of the coordinate.

3 ENABLE PRIVACY IN ASYNCHRONOUS MULTI-TASK LEARNING

The main goal of this paper is to solve a class of regularized multi-task learning (MTL) problems in an asynchronous distributed fashion with a differentially private model in each task node. We first revisit the regularized MTL and analyze the distributed procedure. Then, the necessity of the differential privacy is investigated and the proposed distributed optimization algorithm with privacy guarantee is presented.

3.1 Regularized Multi-Task Learning Revisit

The most fundamental and important assumption of MTL is the tasks relatedness. MTL algorithms transfer knowledge among these tasks using a learned shared representation. Assume that we have T supervised learning tasks in total. For each task t, we are given a training dataset $\mathcal{D}_t = \{X_t, \mathbf{y}_t\}$, where $X_t \in \mathbb{R}^{n_t \times d}$ is the data matrix with feature dimensionality d, $\mathbf{y}_t \in \mathbb{R}^{n_t}$ is the corresponding label vector, and $n_t = |\mathcal{D}_t|$ is the number of data points in task t. For each task, we are interested in learning a model $f(\mathbf{x}; \mathbf{w}) : \mathbb{R}^d \to \mathbb{R}$ from the training data that predicts labels y given a feature vector $\mathbf{x}$, where $\mathbf{w}$ is the set of parameters of the function. In this paper, we assume that we use linear models, therefore $f(\mathbf{x}; \mathbf{w}) = \mathbf{x}^T\mathbf{w}$ and $\mathbf{w}_t \in \mathbb{R}^d$. Based on a specific learning problem, we define a loss function $\ell(f(\mathbf{x}; \mathbf{w}), y)$ to measure the difference between observation and its prediction (e.g., squared loss for regression and logistic loss for classification). We use the notation $\ell_{t,i}(\mathbf{w}_t) = \ell(f(\mathbf{x}_{t,i}; \mathbf{w}_t), y_{t,i})$ to denote the loss for the ith sample of task t. Let $W = [\mathbf{w}_1, \ldots, \mathbf{w}_T] \in \mathbb{R}^{d \times T}$ be the model matrix whose ith column is the task model $\mathbf{w}_t$. Regularized MTL solves the following minimization problem:

$$\min_W \left\{ \sum\nolimits_{t=1}^{T} \left(\frac{1}{n_t} \sum\nolimits_{i=1}^{n_t} \ell_{t,i}(\mathbf{w}_t) \right) + \lambda r(W) \right\}, \tag{1}$$

where $r(\mathbf{W})$ is the regularization term that couples the tasks according to the task relatedness assumption, and λ is a tunable parameter that typically specifies how much knowledge is to be transferred. Examples of commonly used MTL regularizations include shared feature subspace via low-rank modeling [2, 11] and shared feature subset via group sparsity [12, 31, 46].

Directly exploring the task relatedness in W of Equation (1) is sometimes too restrictive. One general class of regularized MTL decomposes the parameters into two components: a shared component and a task specific component. This approach provides the flexibility to trade off between the shared one and the task specific one. For task t, the decomposition is $\mathbf{w}_t = \mathbf{p}_t + \mathbf{q}_t$, where $\mathbf{p}_t \in \mathbb{R}^d$ is the shared component and $\mathbf{q}_t \in \mathbb{R}^d$ is the task specific component. We can collectively express $W = P + Q$, where columns of $P \in \mathbb{R}^{d \times T}$ and $Q \in \mathbb{R}^{d \times T}$ represent respective task components. Thus, the formulation in Equation (1) becomes:

$$\min_{P,Q} \left\{ \sum\nolimits_{t=1}^{T} \left(\frac{1}{n_t} \sum\nolimits_{i=1}^{n_t} \ell_{t,i}(\mathbf{p}_t + \mathbf{q}_t) \right) + \lambda r(P) + \tau g(Q) \right\} \tag{2}$$

where $r(P)$ performs the knowledge transfer and $g(Q)$ regulates the model complexity, and λ and τ are tunable regularization parameters. This formulation subsumes Equation (1) as a special case when $\tau \to \infty$ (and thus $Q = 0$). Many advanced MTL formulations follow this structure, such as dirty model [27] (group sparsity on P and element sparsity on Q), alternating structural optimization [1] (low-rank on P and ℓ_2 on Q), robust feature learning [24] (group sparsity on P and task-wise sparsity on Q), and robust multi-task [12] (low-rank on P and task-wise sparsity on Q). In this work, we assume that $g(Q) = \sum_{t=1}^{T} g_t(Q)$ is separable (e.g., the Frobenius norm of Q).

Since the formulation in Equation (2) includes two natural blocks of variables P and Q, the block coordinate descent algorithm [43] can be used, where we alternatively solve one block and fix the other at each iteration:

$$P^+ = \operatorname*{argmin}_P \left\{ \sum\nolimits_{t=1}^{T} \left(\frac{1}{n_t} \sum\nolimits_{i=1}^{n_t} \ell_{t,i}(\mathbf{p}_t + \mathbf{q}_t) \right) + \lambda r(P) \right\} \tag{3}$$

$$Q^+ = \operatorname*{argmin}_Q \left\{ \sum\nolimits_{t=1}^{T} \left(\frac{1}{n_t} \sum\nolimits_{i=1}^{n_t} \ell_{t,i}(\mathbf{p}_t^+ + \mathbf{q}_t) + \tau g_t(Q) \right) \right\}, \tag{4}$$

where P^+, $\mathbf{p}_t^+$, and Q^+ denote the updated variable blocks. Since the choice of the regularization $r(P)$ is typically non-smooth in MTL problems, (e.g., nuclear norm to induce low-rank, and grouped ℓ_1 norm to induce group sparsity), the update of P^+ requires proximal gradient methods (e.g.,[5, 44]), in which the key sub-procedure is to solve the following proximal operator:

$$\text{prox}_r^{\mu}(X) = \text{argmin}_W \left\{ \tfrac{1}{2} \|W - X\|_F^2 + \mu r(W) \right\}, \tag{5}$$

where μ is a coefficient obtained by the step size and the regularization parameters, and $r(W)$ is required to be a proper and lower semi-continuous function.

3.2 Distributed Learning for Regularized MTL

When the data $\mathcal{D}_t$ of learning tasks are located in different servers and cannot be transferred across the network due to the limited bandwidth and security, a distributed MTL setting is required. In distributed MTL, a *central server* is utilized to host the shared representation and it serves as a nexus to transfer the knowledge among the tasks. Whereas, *task nodes* are responsible for sending summarized information, e.g., gradient of some blocks, to contribute to the learning of the shared representation. A similar setting was given in [4]. In the MTL setting in Equation (2), we need to maintain the shared component P and the task specific component Q. It is thus natural to maintain the shared component P in the central server, and the columns of Q in the task nodes. Even though the update of the component Q can be distributed since the minimization problem in Equation (4) is decoupled for different task nodes, one can immediately notice that procedures in Equation (3) cannot be effectively distributed. The reason is that the gradient evaluation of P requires data, and hence solving it leads to another level of distribution.

We thus propose an alternative approach to perform proximal gradient jointly on the two blocks of variables. Let us collectively define the two components as $Z = \begin{bmatrix} P \\ Q \end{bmatrix} \in \mathbb{R}^{2d \times T}$ and :

$$f(Z) = f(P, Q) = \sum_{t=1}^{T} \left(\frac{1}{n_t} \sum\nolimits_{i=1}^{n_t} \ell_{t,i}(\mathbf{p}_t + \mathbf{q}_t) \right) + \tau g(Q).$$

The proximal update of Equation (2) is given by:

$$Z^+ = \underset{Z}{\operatorname{argmin}} \frac{1}{2} \left\| Z - \left(Z^- - \alpha \nabla_Z f(Z^-) \right) \right\|_F^2 + \alpha \lambda r(P).$$

Since Frobenius norm has the decomposition property, the update of Z can be decomposed into two steps:

$$Q^+ = Q^- - \alpha \nabla_Q f(Z^-) \tag{6}$$

$$P^+ = \operatorname{prox}_r^{\alpha\lambda}(P^- - \alpha \nabla_P f(Z^-)) \tag{7}$$

We notice that the above procedures are much easier to distribute: Q is decoupled and can be distributed for each task. The task node t receives the current shared component $\mathbf{p}_t^-$ from the central server, computes the gradient $\nabla_{\mathbf{p}_t} f(Z^-)$ (using task data $\mathcal{D}_t$), sends it back to the server, and finally locally updates $\mathbf{q}_t^+$ using its data. We note that the gradient $\nabla_{\mathbf{p}_t} f(Z^-)$ can be computed locally because of the following:

$$\nabla_{\mathbf{p}_t} f(Z) = \nabla_{\mathbf{p}_t} f(\mathbf{p}_t, \mathbf{q}_t) = \nabla_{\mathbf{p}_t} \left\{ \frac{1}{n_t} \sum\nolimits_{i=1}^{n_t} \ell_{t,i}(\mathbf{p}_t + \mathbf{q}_t) + \tau g_i(\mathbf{q}_t) \right\}$$

where the computation only depends on the task data $\mathcal{D}_t$. After all the gradients $[\nabla_{\mathbf{p}_1} f(Z^-), \ldots, \nabla_{\mathbf{p}_T} f(Z^-)]$ are received, the server immediately performs proximal computation as in Equation (7), and then sends the columns to the corresponding task nodes.

We can further improve the distributed algorithm in terms of the following perspectives:

- **Successive over-relaxation.** The convergence of the iterative algorithm in the distributed proximal updates can be improved by using a successive over-relaxation (SOR) or Gauss-Seidel improvement [23], where the gradient information is replaced by $\nabla_P f(P^-, Q^+)$, given that the latest information of Q^+ is already available.
- **Semi-exact block updates.** By carefully studying the differences between the block coordinate descent (BCD) type updates in Equation (3) and Equation (4), and proximal type updates in Equation (6) and Equation (7), we can consider the proximal updates as the inexact version of BCD, where the step in Equation (7) can be considered as a single projected gradient step in Equation (3), and similarly the step in Equation (6) can be considered as a single gradient step in Equation (4). While we cannot change the updates of P because of maintaining the efficiency of the distributed structure, we can use the exact version of Q since there is no extra network traffic overhead.
- **Asynchronous update.** Our previous efforts in asynchronous MTL has shown a significant improvement on convergence, and is much more robust against network environments with high and asymmetric latency [4]. In this work, we use the similar framework to allow asynchronous updates, where the server starts to perform proximal update immediately upon receiving $\nabla_{\mathbf{p}_t} f$ from one task node, without waiting for other task nodes. Convergence results on proximal methods have been previously studied in [13, 37].

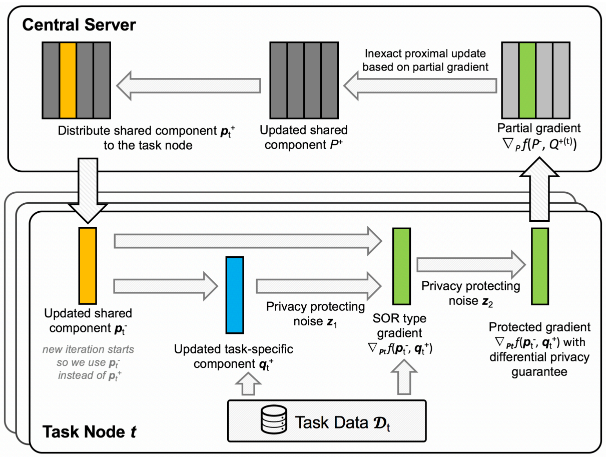

Central Server
Inexact proximal update based on partial gradient
Distribute shared component $\mathbf{p}_t^+$ to the task node
Updated shared component P^+
Partial gradient $\nabla_P f(P^-, Q^{+(t)})$
Updated shared component $\mathbf{p}_t^-$
new iteration starts so we use $\mathbf{p}_t^-$ instead of $\mathbf{p}_t^+$
Updated task-specific component $\mathbf{q}_t^+$
Privacy protecting noise $\mathbf{z}_1$
SOR type gradient $\nabla_{\mathbf{p}_t} f(\mathbf{p}_t^-, \mathbf{q}_t^+)$
Privacy protecting noise $\mathbf{z}_2$
Protected gradient $\nabla_{\mathbf{p}_t} f(\mathbf{p}_t^-, \mathbf{q}_t^+)$ with differential privacy guarantee
Task Data $\mathcal{D}_t$
Task Node t

Figure 1: Overview of the proposed distributed multi-task learning framework with asynchronous updates and privacy protection.

With the aforementioned improvements, the algorithm can be described as follows:

$$\mathbf{q}_t^+ = \underset{\mathbf{q}_t}{\operatorname{argmin}} \left\{ \frac{1}{n_t} \sum\nolimits_{i=1}^{n_t} \ell_{t,i}(\mathbf{p}_t^+ + \mathbf{q}_t) + \tau g_t(\mathbf{q}_t) \right\} \tag{8}$$

$$P^+ = \operatorname{prox}_r^{\alpha\lambda}(P^- - \alpha \nabla_P f(P^-, Q^{+(t)})) \tag{9}$$

where $Q^{+(t)} = [\mathbf{q}_1^-, \ldots, \mathbf{q}_t^+, \ldots, \mathbf{q}_T^-]$ is the partially updated block corresponding to t-th task model. Because of the separability discussed above, the partial gradient requires only the updated gradient $\nabla_{\mathbf{p}_t} f(\mathbf{p}_t^-, \mathbf{q}_t^+)$ of the t-th task node:

$$\nabla_P f(P^-, Q^{+(t)}) = [\nabla_{\mathbf{p}_1} f(\mathbf{p}_1^-, \mathbf{q}_1^-), \ldots, \nabla_{\mathbf{p}_t} f(\mathbf{p}_t^-, \mathbf{q}_t^+), \ldots].$$

3.3 Privacy Issues in Distributed MTL

Using private and sensitive data such as medical and financial records raises privacy issues in data mining. In our framework, the gradient $\nabla_{\mathbf{p}_t} f(\mathbf{p}_t^-, \mathbf{q}_t^+)$ sent to the central server may contain sensitive information from the task node t, and the gradient protocol may lead to privacy leakage. As the gradients are linear transformations of the data matrix, it is also possible to recover the original data after collecting enough gradient vectors with controlled inputs (i.e., $\mathbf{p}_t, \mathbf{q}_t$ in our case), by using sensing techniques [3]. Also, the shared component $\mathbf{p}_t$ sent back to the task node t contains sensitive information from the other task nodes, due to the knowledge transfer procedure (proximal mapping) at the central server. In our setting, we assume that the observer locates at each task node and tries to hijack the message sent from central server.

In this paper, our goal is to protect the *differential privacy* of each data point in each task node (formally defined in Definition 3.1). Differential privacy [16] provides a quantifiable level of privacy with respect to the individual data points. The differential privacy makes sure that the adversary's prior and posterior views about an individual (i.e. before and after having access to the database) shouldn't be "drastically different" or the access to the database shouldn't change the adversary's views about any individual "too much" [17, 30]. If nothing is learned about an individual, then the individual cannot be harmed by the analysis [19]. Algorithms that guarantee differential privacy are randomized by adding noise before, during or after computing functions of the data [18, 19]. In such cases, the differential privacy is the property of the algorithm, not the property of data point or the database. Note that the aforementioned differential privacy is not an algorithm on its own either. Furthermore, it was also proved that the differential privacy is much more efficient than "ad hoc" solutions such as anonymization of patient information [36].

3.4 Privacy-Preserving Distributed MTL

In this section, we discuss the proposed algorithm for the asynchronous distributed MTL with differential privacy. We first review a few key definitions. We denote an algorithm with privacy property by $A_p(\cdot)$, which is randomized so that the re-identification of the data on the user's side is very difficult (by the observer). Formally:

Definition 3.1 (ϵ-differential privacy [16]). An algorithm $\mathcal{A}_p$ is ϵ-differentially private if for any subset of outputs $\mathbf{S}$:

$$\mathbb{P}(\mathcal{A}_p(\mathcal{D}) \in \mathbf{S}) \leq e^{\epsilon} \cdot \mathbb{P}(\mathcal{A}_p(\mathcal{D}') \in \mathbf{S}), \tag{10}$$

for any database $\mathcal{D}$ and $\mathcal{D}'$ differing in a single point, where the $\mathcal{A}_p(\mathcal{D})$ and $\mathcal{A}_p(\mathcal{D}')$ are the outputs of the algorithm for input databases $\mathcal{D}$ and $\mathcal{D}'$, respectively, and $\mathbb{P}$ is the randomness of the noise in the algorithm. □

It can be shown that this is equivalent to $\left|\log\left(\frac{P(\mathcal{A}_p(\mathcal{D})=s)}{P(\mathcal{A}_p(\mathcal{D}')=s)}\right)\right| \leq \epsilon$ for some point s in the output range. Privacy parameter ϵ is used to determine the privacy level. For instance, a small ϵ (≤ 0.1) means that the output probabilities of $\mathcal{D}$ and $\mathcal{D}'$ at s are very similar to each other, which indicates higher noise and hence more privacy, and vice versa. And we use $\epsilon = \infty$ as the non-private case. In our distributed setting, we assume an untrustworthy central server (with semi-malicious adversary), which means that at the central server there may be an observer trying to learn the property of data points in certain task node using the information sent by the corresponding node. Observer may also be the task node's itself which tries to analyze the other task nodes by the information received from central node. Therefore, in each iteration, a differentially private $\mathbf{q}_t$ at the task node and a differentially private $\mathbf{P}$ at the central server should be produced to avoid privacy violations by any observer.

Another commonly used concept is (ϵ, δ)-differential privacy:

Definition 3.2 ((ϵ, δ)-differential privacy). An algorithm $\mathcal{A}_p$ is (ϵ, δ)-differentially private if for any subset of outputs $\mathbf{S}$:

$$\mathbb{P}(\mathcal{A}_p(\mathcal{D}) \in \mathbf{S}) \leq e^{\epsilon} \cdot \mathbb{P}(\mathcal{A}_p(\mathcal{D}') \in \mathbf{S}) + \delta, \tag{11}$$

where δ measures the violation of "pure" differential privacy. □

Similarly it is equivalent to $\left|\log\left(\frac{P(\mathcal{A}_p(\mathcal{D})=s)}{P(\mathcal{A}_p(\mathcal{D}')=s}\right)\right| \leq \epsilon$ with probability $1-\delta$. In this definition, there exists a small output range associated with probability δ such that for fixing a point s in this area, no matter what value ϵ it is, one can always find a pair of databases $\mathcal{D}$ and $\mathcal{D}'$ so that the inequality $\left|\log\left(\frac{P(\mathcal{A}_p(\mathcal{D})=s)}{P(\mathcal{A}_p(\mathcal{D}')=s}\right)\right| \geq \epsilon$ hold.

According to [19], an uncertainty is introduced in the response to be able to hide the participation of an individual. This is quantified by sensitivity, measuring the amount of the change in the function $\mathcal{A}$ made by a single data point in the worst case:

Definition 3.3 (L_2-sensitivity). The L_2-sensitivity of a vector-valued function $\mathcal{A}$ is defined as:

$$S(\mathcal{A}) = \max_i \max_{z_1,\ldots,z_n,z_i'} \left\|\mathcal{A}(z_1,\ldots,z_i,\ldots) - \mathcal{A}(z_1,\ldots,z_i',\ldots)\right\|_2,$$

which is the maximum change in the L_2 norm of the value of the function when one inputs changes. □

Another useful tool for our analysis is the sequential composability theorem. The idea behind this theorem is that if we have $k \geq 2$ algorithms which are independently differentially private, we may feed the next algorithm by the results of the previous ones without a complete loss of privacy. Sequential composability is defined in Theorem 3.4.

Theorem 3.4. *Suppose we have k algorithms $\mathcal{A}_i(\mathcal{D}; z_i)$, $i = 1, \cdots k$, where $\mathcal{D}$ is the dataset and z_i is some auxiliary input. Assume that each of $\mathcal{A}_i$ is ϵ-differentially private for any z_i. Consider a sequence of computations $\{z_1 = \mathcal{A}_1(\mathcal{D}), z_2 = \mathcal{A}_2(\mathcal{D}; z_1), z_3 = \mathcal{A}_3(\mathcal{D}; z_1, z_2), \cdots\}$ and the last one output $z_k = \mathcal{A}(\mathcal{D})$. We have $\mathcal{A}(\mathcal{D})$ guarantee $k\epsilon$-differential privacy.*

Our mechanism is inspired by Chaudhuri *et al.* [9], where the noise vector is drawn from the following multi-dimensional distribution:

$$P(\mathbf{z}) = \frac{1}{\alpha} e^{-\beta\|\mathbf{z}\|}, \tag{12}$$

where $\beta = \frac{\epsilon}{S}$, S is the L_2-sensitivity of the algorithm and α is the normalization constant. Authors of [9] proved that under certain conditions this output perturbation mechanism will guarantee differential privacy.

The privacy issue in stochastic gradient descent scenario needs to take the sequential composition effect into consideration [19], where the privacy leakage accumulates with the number iterations. The noise, hence, should be carefully added so that it cannot be too small (no privacy guaranteed) or too large (very low accuracy). Hence we use two levels of perturbation to guarantee privacy while balance privacy and utility. We summarize the above idea in Algorithm 1.

The S_1 and S_2 mentioned in Algorithm 1 are specified in Lemma 1 and Lemma 2. The algorithm can also be terminated when $\|\mathbf{W}^{k+1} - \mathbf{W}^k\| \leq \nu$ for a small value ν. The bottleneck of the communication speed can be dramatically reduced by sending ∇_t^k (short for $\nabla_{\mathbf{p}_t} f(\mathbf{p}_t^-, \mathbf{q}_t^+)$) to the central server without waiting for the other task nodes. Notice that in gradient descent, if the objective function is L-Lipschitz continuous, the step size should satisfy $\alpha_1^k < \frac{1}{L}$ in order to guarantee convergence.

First, we prove the privacy guarantee. Theoretical guarantee shows that Algorithm 1 protects the privacy of each data point in each task node in a differentially private way.

Lemma 1. *If the norm of all data points satisfies $\|\mathbf{x}_{t,i}\| \leq C_1$. Loss function ℓ is convex, differentiable and $|\ell'(z)| \leq C_2$ for all z. Then $\mathbf{q}_t$ (line 10 in Algorithm 1) guarantees ϵ_1-differential privacy.*

Algorithm 1 Differentially Private Distributed Asynchronous Multi-Task Learning with Untrustworthy Central Server

```
Require: $T$ databases $\{\mathbf{X}_t, \mathbf{y}_t\}$, $t = 1, \cdots, T$ with corresponding loss function $\ell_t$. Parameters $\alpha_1$, $\alpha_2$, $\lambda$, $\epsilon$, total number of iteration MAX.
Ensure: Differentially private models of each task $\mathbf{w}_1, \ldots, \mathbf{w}_T$.
  Initialize $\mathbf{q}_t^{(0)}$ in each task node using STL, $\mathbf{S}^{(0)} = \mathbf{P}^{(0)} = 0_{d\times T}$ on central server side.
  In task node side, for every task node $t$, asynchronously:
  for $k = 1, \ldots, $ MAX do
    Receive $\mathbf{p}_t^{(k-1)}$ from central server.
    Denote $L(\mathbf{q}_t) = \frac{1}{n_t}\sum_{i=1}^{n_t} \ell_{t,i}(\mathbf{p}_t^{(k-1)} + \mathbf{q}_t; \mathbf{x}_{t,i}, \mathbf{y}_{t,i}) + \frac{\lambda}{2}\|\mathbf{q}_t\|_2^2$.
    Update $\mathbf{q}_t$: $\mathbf{q}_t^{(k)} = \mathbf{q}_t^{(k-1)} - \alpha_1^{(k-1)}\left[\nabla_{\mathbf{q}_t} L(\mathbf{q}_t^{(k-1)})\right]$.
    Draw two independent noise vectors $\mathbf{z}_1^k$ and $\mathbf{z}_2^k$ from distribution 12 with $\beta_1 = \frac{\epsilon_1}{S_1}$ and $\beta_2 = \frac{\epsilon_2}{S_2}$.
    Compute $\boldsymbol{\nabla}_t^{(k)} = \frac{1}{n_t}\sum_{i=1}^{n_t} \nabla_{\mathbf{p}_t}\ell_{t,i}(\mathbf{p}_t^{(k-1)} + \mathbf{q}_t^{(k)} + \mathbf{z}_1^k; \mathbf{x}_{t,i}, \mathbf{y}_{t,i})$.
    Compute $\boldsymbol{\nabla}_t^{(k)} = \boldsymbol{\nabla}_t^{(k)} + \mathbf{z}_2^k$ and send $\boldsymbol{\nabla}_t^{(k)}$ to central server without waiting for other tasks.
    In central server side :
    Receive $\boldsymbol{\nabla}_t^{(k)}$, build a matrix $\mathbf{S}^{(k)} \in R^{d\times T}$ such that $\mathbf{S}^{(k)}\Big|_j = \boldsymbol{\nabla}_t^{(k)}$, if$j = t$ and $\mathbf{S}^{(k)}\Big|_j = \mathbf{S}^{(k-1)}\Big|_j$ for$j \neq t$.
    Compute $\mathbf{P}^{(k)} = \mathrm{prox}_{\alpha_2\lambda\|\cdot\|_*}(\mathbf{P}^{(k-1)} - \alpha_2\mathbf{S}^{(k)})$.
    Send $\mathbf{p}_t^{(k)}$ back to corresponding task $t$.
  end for
  for task t, $t = 1, \ldots, T$ do
    $\mathbf{w}_t = \mathbf{p}_t + \mathbf{q}_t$.
  end for
  return $\mathbf{W}$.
```

Proof. First we bound the change of $\mathbf{q}_t$ when one data point changes. This gives the L_2-sensitivity of $\mathbf{q}_t$ and the automatic guarantee of privacy by the analysis in Chaudhuri *et al.* [9]. Without loss of generality we assume that the last data point in $\{X_t, \mathbf{y}_t\}$ is changed from $\mathbf{x}_{t,n_t}, \mathbf{y}_{t,n_t}$ to $\mathbf{x}'_{t,n_t}, \mathbf{y}'_{t,n_t}$, and we have:

$$
\begin{aligned}
&\left\|\mathbf{q}_t^{(k)} - \mathbf{q}_t^{(k)'}\right\|_2\\
&= \|\alpha_1^{(k-1)}\nabla_{\mathbf{q}_t} L'(\mathbf{q}_t^{(k-1)}) - \alpha_1^{(k-1)}\nabla_{\mathbf{q}_t} L(\mathbf{q}_t^{(k-1)})\|_2\\
&= \frac{\alpha_1^{(k-1)}}{n_t}\|\nabla_{\mathbf{q}_t}\ell_{t,n_t}(\mathbf{p}_t^{(k-1)} + \mathbf{q}_t^{(k-1)}; \mathbf{x}'_{t,n_t}, \mathbf{y}'_{t,n_t})\\
&\quad - \nabla_{\mathbf{q}_t}\ell_{t,n_t}(\mathbf{p}_t^{(k-1)} + \mathbf{q}_t^{(k-1)}; \mathbf{x}_{t,n_t}, \mathbf{y}_{t,n_t})\|_2\\
&= \frac{\alpha_1^{(k-1)}}{n_t}\|\ell'_{t,n_t}((\mathbf{p}_t^{(k-1)} + \mathbf{q}_t^{(k-1)})^T\mathbf{x}'_{t,n_t}\mathbf{y}'_{t,n_t})\mathbf{x}'_{t,n_t}\mathbf{y}'_{t,n_t}\\
&\quad - \ell'_{t,n_t}((\mathbf{p}_t^{(k-1)} + \mathbf{q}_t^{(k-1)})^T\mathbf{x}_{t,n_t}\mathbf{y}_{t,n_t})\mathbf{x}_{t,n_t}\mathbf{y}_{t,n_t}\|_2\\
&\le \frac{2\alpha_1^{(k-1)}C_1C_2}{n_t} = S_1,
\end{aligned}
$$

where the last inequality follows the assumption that $|l'(z)| \le C_2$. Therefore, by construction, drawing noise vectors $\mathbf{z}_1^k$ from distribution in Equation 12 with $\beta = (\epsilon_1 n_t)/(2\alpha_1^{(k-1)}C_1C_2)$ guarantees ϵ-differential privacy with respect to the data set $\{\mathbf{X}_t, \mathbf{y}_t\}$. □

Lemma 2. *With the same assumptions in Lemma 1, $\boldsymbol{\nabla}_t^{(k)}$ (line 13 in Algorithm 1) guarantees ϵ_2-differential privacy with fixed auxiliary input $\mathbf{p}_t^{(k-1)} + \mathbf{q}_t^{(k)} + \mathbf{z}_1^k$.*

Proof. We compute the L_2-sensitivity of $\boldsymbol{\nabla}_t^{(k)}$,

$$
\begin{aligned}
&\left\|\boldsymbol{\nabla}_t^{(k)} - \boldsymbol{\nabla}_t^{(k)'}\right\|_2\\
&= \frac{1}{n_t}\|\nabla_{\mathbf{p}_t}\ell_{t,n_t}(\mathbf{p}_t^{(k-1)} + \mathbf{q}_t^{(k)} + \mathbf{z}_1^k; \mathbf{x}_{t,n_t}, \mathbf{y}_{t,n_t})\\
&\quad - \nabla_{\mathbf{p}_t}\ell_{t,n_t}(\mathbf{p}_t^{(k-1)} + \mathbf{q}_t^{(k)} + \mathbf{z}_1^k; \mathbf{x}'_{t,n_t}, \mathbf{y}'_{t,n_t})\|_2\\
&= \frac{1}{n_t}\|\ell'_{t,n_t}((\mathbf{p}_t^{(k-1)} + \mathbf{q}_t^{(k)} + \mathbf{z}_1^k)^T\mathbf{x}_{t,n_t}\mathbf{y}_{t,n_t})\mathbf{x}_{t,n_t}\\
&\quad - \ell'_{t,n_t}((\mathbf{p}_t^{(k-1)} + \mathbf{q}_t^{(k)} + \mathbf{z}_1^k)^T\mathbf{x}'_{t,n_t}\mathbf{y}'_{t,n_t})\mathbf{x}'_{t,n_t}\|_2\\
&\le \frac{2C_1C_2}{n_t} = S_2.
\end{aligned}
$$

Therefore, by construction, drawing noise vectors $\mathbf{z}_2^k$ from the distribution in Equation 12 with $\beta = \frac{\epsilon_2 n_t}{2C_1C_2}$ guarantees ϵ_2-differential privacy with respect to the data set $\left\{\mathbf{X}_t, \mathbf{y}_t\right\}$ when fixing $\mathbf{p}_t^{(k-1)} + \mathbf{q}_t^{(k)} + \mathbf{z}_1^k$. □

Theorem 3.5. *$\boldsymbol{\nabla}_t^{(k)}$ guarantees $\epsilon_1 + \epsilon_2$-differential privacy with respect to the data set $\{\mathbf{X}_t, \mathbf{y}_t\}$ for classification problem with $\mathbf{y}_{t,n_t} \in \{+1, -1\}$.*

Proof. Combine Lemma 1 and Lemma 2 and use Theorem 3.4, we can get a result immediately. □

We proved that the output from any task node guarantees $\epsilon_1 + \epsilon_2$-differential privacy. This shows that sequential composition of differentially private procedures reduces the privacy level in a linear sense. The result can be easily generalized to other loss functions. For regression problem, the loss function is not bounded automatically. Hence we provide the following theorem:

Theorem 3.6. *$\boldsymbol{\nabla}_t^{(k)}$ guarantees $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-differential privacy with respect to the data set $\{\mathbf{X}_t, \mathbf{y}_t\}$ for regression problem with $\mathbf{y}_{t,n_t} \in [+1, -1]$.*

Proof. Here we use least square loss as an example,

$$
\begin{aligned}
&\left\|\mathbf{q}_t^{(k)} - \mathbf{q}_t^{(k)'}\right\|_2\\
&\quad = \|\alpha_1^{(k-1)}\nabla_{\mathbf{q}_t} L'(\mathbf{q}_t^{(k-1)}) - \alpha_1^{(k-1)}\nabla_{\mathbf{q}_t} L(\mathbf{q}_t^{(k-1)})\|_2\\
&\quad = \frac{2\alpha_1^{(k-1)}}{n_t}\left\|\left[(\mathbf{p}_t^{(k-1)} + \mathbf{q}_t^{(k-1)})^T\mathbf{x}_{t,n_t} - \mathbf{y}_{t,n_t}\right]\mathbf{x}_{t,n_t}\right.\\
&\qquad \left. - \left[(\mathbf{p}_t^{(k-1)} + \mathbf{q}_t^{(k-1)})^T\mathbf{x}'_{t,n_t} - \mathbf{y}'_{t,n_t}\right]\mathbf{x}'_{t,n_t}\right\|_2.
\end{aligned}
$$

Here we assume that the $\|\mathbf{p}_{t-1} + \mathbf{q}_{t-1}\|$ is bounded by some constant C_3 with probability δ_1, here the probability is with respect to the data distribution, and the constant pair (C_3, δ_1) is uniquely determined by the data with δ_1 between 0 and 1. By triangle and

Cauchy-Schwarz inequalities, we have the following with probability δ_1:

$$\begin{aligned}
&\left\|\mathbf{q}_t^{(k)} - \mathbf{q}_t^{(k)'}\right\|_2 \\
&\quad\le \frac{2\alpha_1^{(k-1)}}{n_t}\left\|\left[(\mathbf{p}_t^{(k-1)} + \mathbf{q}_t^{(k-1)})^T\mathbf{x}_{t,n_t} - \mathbf{y}_{t,n_t}\right]\right\| \|\mathbf{x}_{t,n_t}\|_2 \\
&\quad+ \left\|\left[(\mathbf{p}_t^{(k-1)} + \mathbf{q}_t^{(k-1)})^T\mathbf{x}'_{t,n_t} - \mathbf{y}'_{t,n_t}\right]\right\| \|\mathbf{x}'_{t,n_t}\|_2 \\
&\quad\le \frac{4\alpha_1^{(k-1)}C_1(C_1C_3+1)}{n_t}.
\end{aligned}$$

Similarly for L_2-sensitivity of $\nabla_t^{(k)}$, we have for another constant pair (C_4, δ_2) such that with probability δ_2:

$$\left\|\nabla_t^{(k)} - \nabla_t^{(k)'}\right\|_2 \le \frac{4C_1(C_1C_4+1)}{n_t}.$$

Hence by construction, with probability $1 - \delta_1 - \delta_2$, our algorithm guarantees $\epsilon_1 + \epsilon_2$-differential privacy. □

Besides the privacy issue, in real world communication, network delay is a frequently encountered issue which is important and needs to be resolved. This is especially critical in a distributed learning system if the delay affects the performance of the task nodes. The problem of the long latency decreases the speed of the whole network, which can be solved by the asynchronous mechanism in the central server. But the performance of the task node with long latency is also affected, which is beyond the ability of the asynchronous mechanism.

Notice that real application involves many iterations and possibly more tasks, the cumulative privacy loss can grow quickly and the privacy can easily become meaningless. This can be reduced by randomly sampling protion of data point from each task set in each iteration and update gradient using these subset instead of whole data set. Privacy amplification theorem [6] implies that each iteration is $(O(q\epsilon), q\delta)$-differentially private with respect to the whole data set where q is the sampling ratio.

Finally, notice that no noise is needed for the output of the central server with the presence of an untrustworthy central server. In the first iteration, the sensitivity of the updated model for task i (output from central server) with respect to task $j \neq i$ is 0 due to the initialization of $\mathbf{P}^{(0)}$, which has nothing to do with the dataset of the task $j \neq i$. In the second iteration, the sensitivity of the updated model for another task $k \neq i$ with respect to the task $j \neq k$ and $j \neq i$ is 0, because of the same reason as in the first iteration. The privacy of ith dataset has been already protected due to the privacy-preserving gradient sent by the task i and the post-processing principle mentioned in [19]. Hence, in the mth iteration, the output model of the central server is privacy-preserved due to initialization and privacy-preserved information in the previous iterations less than m. However, this does not hold in the case of trustworthy central server, where the data set i is safe with respect to central server but not any task node $j \neq i$.

4 EXPERIMENTS

In this section, we present the detailed experiments conducted to investigate the convergence properties of the proposed Algorithm 1, which is implemented in Java. Furthermore, the performance of each task node is tested with and without differential privacy. We also investigated the relationship between the values of the centralized objective function given in Equation 1 and the privacy parameter ϵ. Finally, we observed how the model performance of each task node changes with ϵ. Our experiments showed that the proposed algorithm performs well on both classification and regression problems for non-private and private cases.

Dataset Details	Synthetic	ADNI
Number of tasks	20	20
Sample size of each task	200	15-27
Data dimensionality	28	248
Labels	c/r	c/r

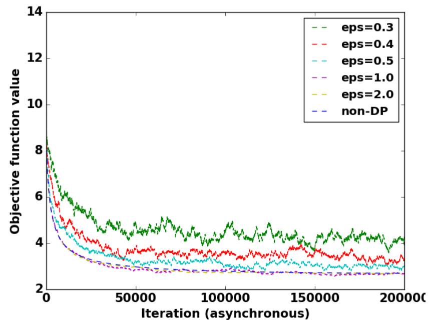
Table 1: Details of the datasets used in the experiments. The ADNI's datasets are obtained from the Alzheimer's Disease Neuroimaging Initiative (http://adni.loni.usc.edu/). The total number of tasks in ADNI datasets is 79. We selected 20 of them as our datasets.

4.1 How Does Privacy Level Affect the Convergence of the Objective Function.

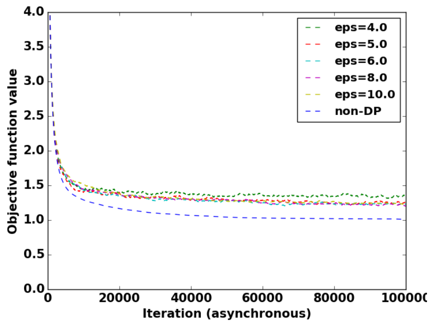
In this experiment, we empirically show the relationship between the value of the centralized objective function in Equation 1 and different ϵ values with respect to the number of iterations. Experiments on both synthetic and real world datasets in Table 1 demonstrate that our asynchronous algorithm can converge to the optimal solution. Furthermore, it is worth noting that the privacy requirements inevitably degrade classifier performance, which is one of the most important trade-off phenomena in differentially private algorithm.

In Table 1, c/r means that the dataset can be used in either classification problems or regression problems. For all the task nodes and datasets, we split 30% of data for training and the rest for testing. The number of task nodes in our system is fixed to 20 and the latency of each task is randomly chosen between 0-15 milliseconds to simulate the network delays. The regularization parameter λ is set to 10^{-3} for the classification problem with logistic loss and 10^{-5} for the regression problem with least square loss, using 10-fold cross validation. In Figures 2a, 2b and 2c, $\alpha_1 = \alpha_2 = 10^{-1}$ and $\mu = 1.5 \times 10^{-3}$. In Figure 7, $\alpha_1 = \alpha_2 = 10^{-3}$ and $\mu = 1.0 \times 10^{-4}$. These settings ensure that the rank of $\mathbf{P}^{(k)}$ at the central server is 3 in most time. Hence the task models are projected into a low rank subspace and the information of tasks is transferred [49]. Proximal mapping at the central server can be performed by singular value decomposition, which is given in Theorem 3.1 in [28]. Number of iterations is set to $10,000$ for all the task nodes.

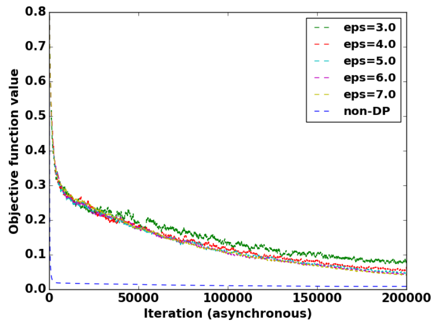
In Figure 2, we plot the value of the objective function with respect to the number of iterations for different datasets and learning problems. In each figure, multiple curves are plotted with different ϵ values. In Figure 2a and Figure 2c, $\epsilon_1 = \epsilon_2 = \epsilon$. In Figure 2b, $\epsilon_1 = 0.5$, ϵ_2 is the ϵ value shown in the figure, which shows that the first level of noise has very limited impact on the performance. In Figure 7, $\epsilon_1 = 10$. It can be seen that the proposed algorithm converges for both differentially private and non-private cases. The curve converges fast in classification problems (Figure 2a and Figure 2b). The noise can be controlled by ϵ, and a smaller ϵ results in a larger objective value for the same number



(a) Classification for synthetic dataset

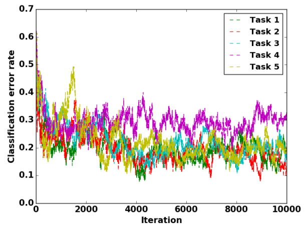


(b) Classification for ADNI dataset

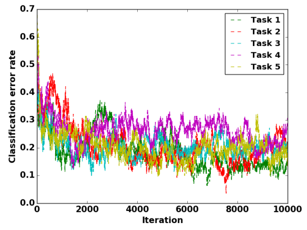


(c) Regression for synthetic dataset

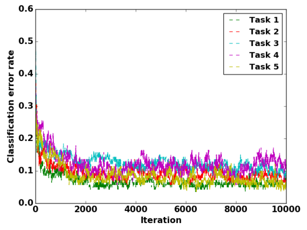
Figure 2: Value of the objective function with respect to the number iterations for different datasets and learning problems with several ϵ values. The proposed algorithm converges for both differentially private and non-private cases.



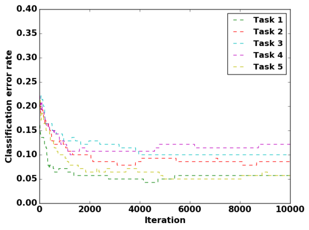
(a) ϵ=0.3



(b) ϵ=0.4

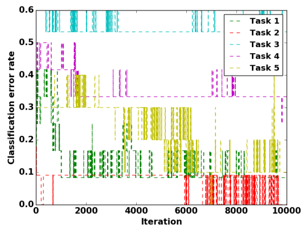


(c) ϵ=2.0

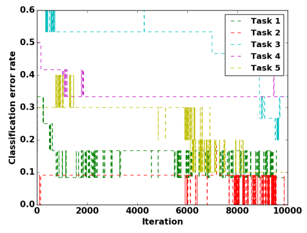


(d) ϵ=∞

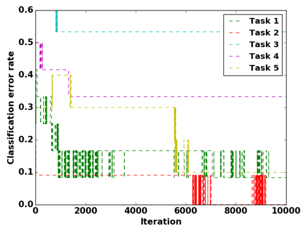
Figure 3: Classification error rate of synthetic dataset.



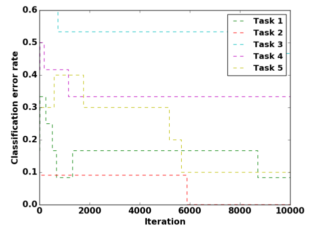
(a) ϵ=4.0



(b) ϵ=6.0

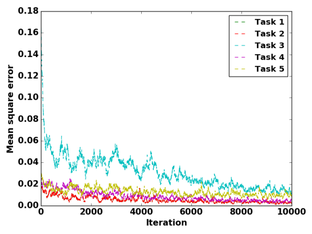


(c) ϵ=10.0

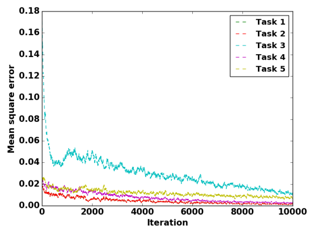


(d) ϵ=∞

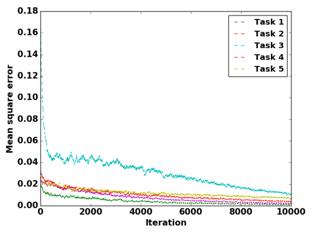
Figure 4: Classification error rate of ADNI dataset.



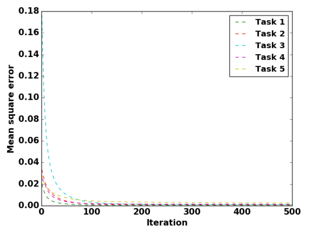
(a) ϵ=3.0



(b) ϵ=5.0



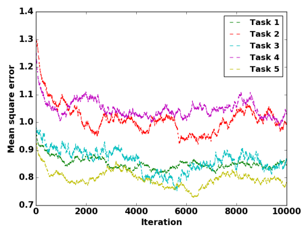
(c) ϵ=7.0

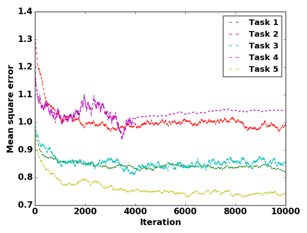


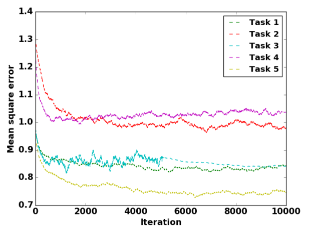
(d) ϵ=∞

Figure 5: Mean square error of regression for the synthetic dataset.

iterations. This observation is due to the perturbation of the noise. The objective value presents more fluctuations as the ϵ gets smaller, which is clearer in the synthetic data set. This also reflects the fact brought by the Definition 3.1 that smaller ϵ introduces more constraints and hence more privacy, because a smaller ϵ makes the


(a) ϵ=30.0


(b) ϵ= 50.0


(c) ϵ= 70.0

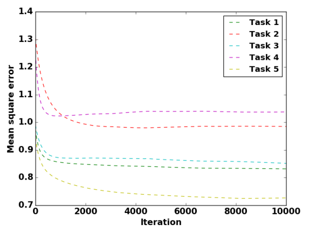

(d) ϵ=∞

Figure 6: Mean square error of regression for the ADNI dataset.

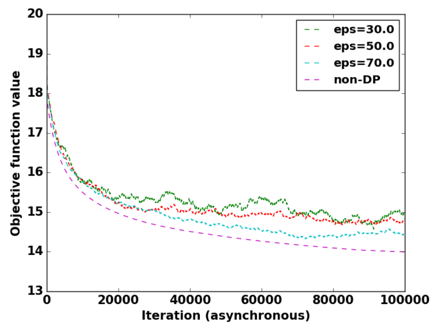


Figure 7: Value of the objective function with respect to the number of iterations, regression for ADNI dataset.

variance of the noise larger and causes more perturbation. Furthermore, Figure 2b shows that the objective value has more tolerance to noise and hence is more stable in terms of the convergence for the ADNI dataset. Figure 2 empirically shows that our algorithm guarantees $\epsilon-$ differential privacy ($(\epsilon, \delta)-$ differential privacy for regression) and a smaller ϵ yields slower convergence because of the noise. As a result, ϵ is recommended to have a value larger than 0.3 to guarantee a good performance and fast convergence. The trade-off between performance and privacy is in the center of differential privacy, as we can also see in these figures. An algorithm providing the same $\epsilon-$level privacy with less noise added is always preferable.

4.2 How Does Privacy Level Affect the Performance of Task Nodes.

The goal of this experiment is to show the change in the performance of the task models with respect to the number of iterations for different ϵ values. Parameter settings are the same as the settings of the experiment presented in Section 4.1 and ϵ is tuned at each task node simultaneously. Performance evaluation of the task models is error rate for the classification problems and mean squared error (MSE) for the regression problems. For each MTL problem, we plot the performance of the first 5 tasks for clarity. In this experiment, each plot in Figures 3, 4, 5 and 6 shows the results for the corresponding plot in Figures 2a, 2b, 2c and Figure 7.

As Seen from Figure 3, despite a rise in the error rate, the variance of the fluctuations goes down as the ϵ gets larger. This is due to the fact that the ϵ controls the noise shape and hence directly affects the performance. These two important characteristics are shared by all figures of private cases. These two characteristics indicate that our algorithm indeed guarantees the differential privacy. The result in Figure 3 is relatively better than that in Figure 4 because we have much lesser data points[2] and higher dimension in Figure 4. But convergence is still achieved under such severe environment with very limited data, noisy and asynchronous update. Datasets with larger sample sizes perform better for the same ϵ since a larger n_t means lower sensitivity and hence needs less noise to mask it. Also in regression problem 5, decrease in ϵ does not increase the MSE dramatically, which is another good property of the proposed algorithm. We need to point out that the smallest ϵ we use in each of the figures in Figure 2 is the lower bound of ϵ that can guarantee the convergence of each of them empirically. This limit may be affected by the data dimension, learning rate and other factors, which is a further research interest. All three figures show that our model converges both in private and non-private cases, which indicates that our proposed method can be widely used in many scenarios.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we propose a new asynchronous MTL algorithm that preserves the privacy of local data in a differentially private sense. Our algorithm is proved to guarantee differential privacy in its strongest form: ϵ-differential privacy in classification problem and a weaker form: (ϵ, δ)-differential privacy in regression problem. We conducted series of experiments to show that our algorithm converges under the conditions of both noisy and lack of data situations. These conditions are very common in real world applications and our approach is applicable to them. For future work, we will consider trustworthy central server where the observer exists only in task node side and the noise is needed only on central server side. We expect that the performance of this situation will be better than the untrustworthy case since only one level noise is added. The noise is not directly added to the gradient either. Next, we will also take into consideration of a long-term privacy accumulating effect. An observer gets increasing amount of information at the central server with the number of iterations and hence the joint distribution of the output gradient is concentrated. This is a unique situation for a gradient descent-based algorithm and it requires careful study and exploration.

[2]This is also the reason of those step-like decreasing.

6 ACKNOWLEDGMENTS

This research is supported in part by National Science Foundation under Grant IIS-1565596, IIS-1615597, and the Office of Naval Research under grant number N00014-14-1-0631 and N00014-17-1-2265.

REFERENCES

[1] Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research* 6, Nov (2005), 1817–1853.

[2] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. 2008. Convex multi-task feature learning. *Machine Learning* 73, 3 (2008), 243–272.

[3] Richard G Baraniuk. 2007. Compressive sensing [lecture notes]. *IEEE signal processing magazine* 24, 4 (2007), 118–121.

[4] Inci M. Baytas, Ming Yan, Anil K. Jain, and Jiayu Zhou. 2016. Asynchronous Multi-task Learning. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. 11–20.

[5] Amir Beck and Marc Teboulle. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences* 2, 1 (2009), 183–202.

[6] Amos Beimel, Shiva Prasad Kasiviswanathan, and Kobbi Nissim. 2010. Bounds on the sample complexity for private learning and private data release. In *Theory of Cryptography Conference*. Springer, 437–454.

[7] Jérôme Bolte, Shoham Sabach, and Marc Teboulle. 2014. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming* 146, 1-2 (2014), 459–494.

[8] Rich Caruana. 1998. Multitask learning. In *Learning to learn*. Springer, 95–133.

[9] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. 2011. Differentially private empirical risk minimization. *The Journal of Machine Learning Research* 12 (2011), 1069–1109.

[10] Kamalika Chaudhuri, Anand D Sarwate, and Kaushik Sinha. 2013. A near-optimal algorithm for differentially-private principal components. *The Journal of Machine Learning Research* 14, 1 (2013), 2905–2943.

[11] Jianhui Chen, Lei Tang, Jun Liu, and Jieping Ye. 2009. A convex formulation for learning shared structures from multiple tasks. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 137–144.

[12] Jianhui Chen, Jiayu Zhou, and Jieping Ye. 2011. Integrating low-rank and group-sparse structures for robust multi-task learning. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 42–50.

[13] Damek Davis. 2016. The Asynchronous PALM Algorithm for Nonsmooth Nonconvex Problems. *arXiv preprint arXiv:1604.00526* (2016).

[14] Damek Davis, Brent Edmunds, and Madeleine Udell. 2016. The Sound of APALM Clapping: Faster Nonsmooth Nonconvex Optimization with Stochastic Asynchronous PALM. *arXiv preprint arXiv:1606.02338* (2016).

[15] Francesco Dinuzzo, Gianluigi Pillonetto, and Giuseppe De Nicolao. 2011. Client–server multitask learning from distributed datasets. *IEEE Transactions on Neural Networks* 22, 2 (2011), 290–303.

[16] Cynthia Dwork. 2006. Differential privacy. In *Automata, languages and programming*. Springer, 1–12.

[17] Cynthia Dwork. 2008. An ad omnia approach to defining and achieving private data analysis. In *Privacy, Security, and Trust in KDD*. Springer, 1–13.

[18] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography*. Springer, 265–284.

[19] Cynthia Dwork and Aaron Roth. 2013. The algorithmic foundations of differential privacy. *Theoretical Computer Science* 9, 3-4 (2013), 211–407.

[20] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. 2010. Boosting and differential privacy. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*. IEEE, 51–60.

[21] Theodoros Evgeniou and Massimiliano Pontil. 2004. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 109–117.

[22] Arik Friedman and Assaf Schuster. 2010. Data mining with differential privacy. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 493–502.

[23] Gene H Golub and Charles F Van Loan. 2012. *Matrix computations*. Vol. 3. JHU Press.

[24] Pinghua Gong, Jieping Ye, and Changshui Zhang. 2012. Robust multi-task feature learning. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 895–903.

[25] Sunil Kumar Gupta, Santu Rana, and Svetha Venkatesh. 2016. Differentially Private Multi-task Learning. In *Pacific-Asia Workshop on Intelligence and Security Informatics*. Springer, 101–113.

[26] Prateek Jain and Abhradeep Thakurta. 2013. Differentially private learning with kernels. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*. 118–126.

[27] Ali Jalali, Sujay Sanghavi, Chao Ruan, and Pradeep K Ravikumar. 2010. A dirty model for multi-task learning. In *Advances in Neural Information Processing Systems*. 964–972.

[28] Shuiwang Ji and Jieping Ye. 2009. An accelerated gradient method for trace norm minimization. In *Proceedings of the 26th annual international conference on machine learning*. ACM, 457–464.

[29] Xin Jin, Ping Luo, Fuzhen Zhuang, Jia He, and Qing He. 2015. Collaborating between Local and Global Learning for Distributed Online Multiple Tasks. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 113–122.

[30] Shiva P Kasiviswanathan and Adam Smith. 2014. On the'Semantics' of Differential Privacy: A Bayesian Formulation. *Journal of Privacy and Confidentiality* 6, 1 (2014), 1.

[31] Seyoung Kim and Eric P Xing. 2010. Tree-guided group lasso for multi-task regression with structured sparsity. (2010).

[32] Matt J Kusner, Jacob R Gardner, Roman Garnett, and Kilian Q Weinberger. 2015. Differentially Private Bayesian Optimization.. In *ICML*. 918–927.

[33] Chao Li, Michael Hay, Vibhor Rastogi, Gerome Miklau, and Andrew McGregor. 2010. Optimizing linear counting queries under differential privacy. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 123–134.

[34] Sulin Liu, Sinno Jialin Pan, and Qirong Ho. 2016. Distributed Multi-task Relationship Learning. (2016). arXiv:arXiv:1612.04022

[35] David Mateos-Núñez, Jorge Cortés, and Jorge Cortes. 2015. Distributed optimization for multi-task learning via nuclear-norm approximation. *IFAC-PapersOnLine* 48, 22 (2015), 64–69.

[36] Arvind Narayanan and Vitaly Shmatikov. 2008. Robust de-anonymization of large sparse datasets. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*. IEEE, 111–125.

[37] Zhimin Peng, Yangyang Xu, Ming Yan, and Wotao Yin. 2016. ARock: an algorithmic framework for asynchronous parallel coordinate updates. *SIAM Journal on Scientific Computing* 38, 5 (2016), A2851–A2879.

[38] Arun Rajkumar and Shivani Agarwal. 2012. A Differentially Private Stochastic Gradient Descent Algorithm for Multiparty Classification.. In *AISTATS*. 933–941.

[39] Peter Richtárik and Martin Takáč. 2013. Distributed Coordinate Descent Method for Learning with Big Data. (2013). arXiv:arXiv:1310.2059

[40] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. ACM, 1310–1321.

[41] Abhradeep Guha Thakurta and Adam Smith. 2013. Differentially private feature selection via stability arguments, and the robustness of the lasso. In *Conference on Learning Theory*. 819–850.

[42] Paul M Thompson, Jason L Stein, Sarah E Medland, Derrek P Hibar, Alejandro Arias Vasquez, Miguel E Renteria, Roberto Toro, Neda Jahanshad, Gunter Schumann, Barbara Franke, and others. 2014. The ENIGMA Consortium: large-scale collaborative analyses of neuroimaging and genetic data. *Brain imaging and behavior* 8, 2 (2014), 153–182.

[43] Paul Tseng. 2001. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications* 109, 3 (2001), 475–494.

[44] Stephen J Wright, Robert D Nowak, and Mário AT Figueiredo. 2009. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing* 57, 7 (2009), 2479–2493.

[45] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. 2011. Differential privacy via wavelet transforms. *Knowledge and Data Engineering, IEEE Transactions on* 23, 8 (2011), 1200–1214.

[46] Xiaolin Yang, Seyoung Kim, and Eric P Xing. 2009. Heterogeneous multitask learning with joint sparsity constraints. In *Advances in neural information processing systems*. 2151–2159.

[47] Jiayu Zhou, Jianhui Chen, and Jieping Ye. 2011. Clustered multi-task learning via alternating structure optimization. In *Advances in neural information processing systems*. 702–710.

[48] Jiayu Zhou, Jianhui Chen, and Jieping Ye. 2011. *MALSAR: Multi-tAsk Learning via StructurAl Regularization*. Arizona State University. http://www.MALSAR.org

[49] Jiayu Zhou, Jun Liu, Vaibhav A Narayan, and Jieping Ye. 2012. Modeling disease progression via fused sparse group lasso. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1095–1103.

[50] Jiayu Zhou, Lei Yuan, Jun Liu, and Jieping Ye. 2011. A multi-task learning formulation for predicting disease progression. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 814–822.