Assisted Learning and Imitation Privacy

Xun Xian, Xinran Wang, Jie Ding, Reza Ghanadan

Abstract

Motivated by the emerging needs of decentralized learners with personalized learning objectives, we present an Assisted Learning framework where a service provider Bob assists a learner Alice with supervised learning tasks without transmitting Bob's private algorithm or data. Bob assists Alice either by building a predictive model using Alice's labels, or by improving Alice's private learning through iterative communications where only relevant statistics are transmitted. The proposed learning framework is naturally suitable for distributed, personalized, and privacy-aware scenarios. For example, it is shown in some scenarios that two suboptimal learners could achieve much better performance through Assisted Learning. Moreover, motivated by privacy concerns in Assisted Learning, we present a new notion of privacy to quantify the privacy leakage at learning level instead of data level. This new privacy, named imitation privacy, is particularly suitable for a market of statistical learners each holding private learning algorithms as well as data.

I. Introduction

Rapid developments in communications, networking, robotics, genomics, novel materials, and powerful computation platforms are rapidly bringing data-generating people, processes and devices together. We are in an era where there exists an emerging number of statistical learners, each holding personalized data and domain-specific objective goals. Each learner often needs to integrate data from diverse sources or coordinate with other learners in order to facilitate personalized learning objectives. With a decentralized set of learners, an urgent related issue is to protect learners' privacy with respect to data as well as sophisticated algorithms (or models). The interactions between multiple learners in privacy-aware scenarios motivate us to consider a learning framework where each learner can be assisted by others without transmitting sensitive information, and a related notion of privacy.

From the perspective of Machine-Learning-as-a-Service (MLaaS) [1], [2], existing assistance between learners is mainly in the following scenario. A service provider receives predictor-label pairs (x, y) from data curators and then learn a private supervised model from such data. The service provider then provides prediction services for future data of the data curator or possibly other users, who send future predictor $\tilde{x}$ of a similar nature to inquire a prediction of the corresponding label $\tilde{y}$. Consider a set of learners who collect relevant features from a common population of interest, e.g., a group of patients, a cohort of mobile users, a basket of financial assets, etc. An excelling learner may provide *services* that does not transmit data but still convey information relevant to others' learning objectives.

Suppose that there exist two learners Alice and Bob who collect various features from the same group of people. Alice wants to develop a new product, and she has the labels of interest from each person. Bob holds unique features which Alice does not, and those features may be relevant to Alice's prediction goal. Bob intends to assist Alice, but Bob will not disclose data even if they are reasonably perturbed. Then, a natural way of Bob assisting Alice is to simply receive her labels, collate them with his own private data, and learn a supervised algorithm privately. Bob then provides prediction services for Alice who inquires with future data, for example in the form of an application programming interface (API). Moreover, suppose that Alice also has a private learning algorithm and private data features that can be (partially) collated to Bob's. Is it possible to still benefit from the *algorithm* as well as *data* held by Bob? A classical approach is to for Alice to perform model selection from her own model and Bob's private model (through Bob's API), and then decide whether to use Bob's service in the future. A related approach is to perform statistical model averaging over the two learners. However, neither approach will significantly outperform the better one of Alice and Bob [3], [4]. The above approaches, however, do not fully utilize all the available data which is a union of Alice's and Bob's. Is it possible for Alice to achieve the performance as if all the private information of Alice and Bob were centralized? This motivates us to propose the concept of *Assisted Learning*, where the main idea is to treat labels y as public (to transmit) and predictors x as private (not to transmit). And

X. Xian, X. Wang, J. Ding are with the School of Statistics, University of Minnesota, MN 55414, USA. R. Ghanadan is with Google AI.

TABLE I: Examples of Bob assisting Alice (none of whom will transmit personalized models or data).

Alice	Research group A	Hospital	Mobile device	Investor	EEG
Bob	Research group B	Lab	Cloud service	Financial trader	Eye Movement
Collating Index	Data index	Patient ID	User ID	Time stamp	Subject ID

we show that for a suitably chosen y at each iteration of communications, Alice may benefit from Bob as if she had Bob's data. Some common scenarios of this nature are showed in Tab. I.

In correspondence to the privacy concerns in assisted learning, we also develop a new notion called imitation privacy. The proposed notion of privacy is motivated by a reasonable concern of Bob that his capability to generate algorithms during assisting other is to be imitated, when an adversary Alice keeps querying. Such an imitation, if accurate or near-accurate, could cause a considerable damage to Bob when his core competitive advantage is his black-box learning procedure that includes not only data but also sophisticated algorithms being deployed. This black-box level privacy is different from data level privacy since its focus is on protecting a learner's capability to generate predictive models, instead of the data itself. In other words, Bob's assisted learning service is the object to protect. For instance, in many domains such as financial trading and environmental prediction, data may be easily accessible by many learners but what really matters is an effective algorithm being deployed. We will show examples where imitation privacy and data privacy (in particular differential privacy [5]–[7]) do not imply each other.

Our main contributions in this work are three folds:

- We introduce the notion of Assisted Learning that is naturally suitable for a variety of learning scenarios.
- Based on assisted learning, we develop some concrete protocols so that a service provider can assist others by improving their predictive performances. We show that the proposed learning protocol can be applied for a wide range of nonlinear and nonparametric learning tasks, where near-oracle performance can be achieved. Some preliminary results on the oracle performance are developed.
- We propose a concept of privacy that focuses on the protection of service-providing modules (or data-model pairs), and discuss it through concrete examples.

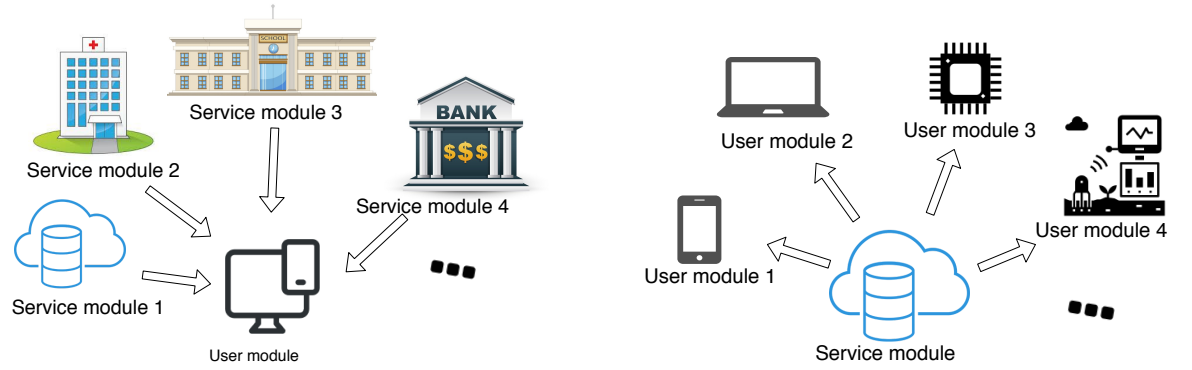
II. Assisted Learning

A. Notation

Throughout the paper, we let $X \in \mathcal{X}^{n\times p}$ denote a general data matrix which consists of n items and p features, and $y \in \mathcal{Y}^n$ be a vector of labels (or responses), where $\mathcal{X}, \mathcal{Y} \subseteq \mathbb{R}$. Let x_i denotes the ith row of X. A supervised function f approximates $x_i \mapsto \mathbb{E}(y_i \mid x_i)$ for a pair of predictor (or feature) $x_i \in \mathcal{X}^p$ and $y_i \in \mathcal{Y}$. Let $f(X)$ denote an $\mathbb{R}^n$ vector whose ith element is $f(x_i)$. We say two matrices or column vectors A, B are *collated* if rows of A and B are aligned with some common index. For example, the index can be date or time stamps for datasets of time series, or personal identification number for datasets of mobile users. Let $\mathcal{N}(\mu, \sigma^2)$ denote the Gaussian distribution with mean μ and standard deviation σ.

B. Supervised Learning with Personalized Services

We first depict how we envision Assisted Learning through a concrete usage scenario. Alice is equipped with a set of labelled data (X_A, Y_A) and a supervised learning algorithm for the data. Some other researchers, say Bob, may be performing different learning tasks with distinct data (X_B, Y_B) and learning models. where X_A and X_B can be (partially) collated. Alice wishes to be assisted by Bob to facilitate her own learning, while maintaining both of their sensitive information. On the other hand, Alice would also be glad to assist others for potential rewards. A set of learning modules such as Alice constitute a statistical learning market where each module can either provide or receive assistance to facilitate personalized learning goals. Figures 1 illustrates Assisted Learning from a user's perspective and a service provider's perspective.



(a) User-centered perspective: a user module sends queries to service modules to obtain feedbacks and improve its personal learning.

(b) Service-centered perspective: a service module assists user modules by delivering feedbacks using its personal data and model.

Fig. 1: Assisted Learning from two perspectives.

C. General Description of Assisted Learning

We first introduce our notions of algorithm and module in the context of supervised learning.

Definition 1 (Algorithm). *A learning algorithm $\mathcal{A}$ is a mapping from a dateset $X \in \mathbb{R}^{n\times p}$ and label vector $y \in \mathbb{R}^n$ to a prediction function $f_{\mathcal{A},X,y} : \mathbb{R}^p \to \mathbb{R}$.*

An algorithm may represent linear regression, ensemble method, neural networks, or a set of models from which a suitable one is chosen using model selection techniques [4], [8]. For example, when the least squares method is used to learn the supervised relation between X and y, then $f_{\mathcal{A},X,y}$ is a linear operator: $\tilde{x} \mapsto \tilde{x}^{\mathrm{T}}(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}y$ for a predictor $\tilde{x} \in \mathbb{R}^p$. The above $f_{\mathcal{A},X,y}$ is also called a hypothesis in some literature of classification.

Definition 2 (Module). *A module $\mathcal{M} = (\mathcal{A}, X)$ is a pair of algorithm $\mathcal{A}$ and observed dateset X. For a given label vector $y \in \mathbb{R}^n$, a module naturally induces a prediction function $f_{\mathcal{A},X,y}$. We simply write $f_{\mathcal{A},X,y}$ as $f_{\mathcal{M},y}$ whenever there is no ambiguity.*

Recall from Subsection II-A that the above X is assumed to be in $\mathbb{R}^{n\times p}$, representing n items and p features. In the context of assisted learning, $\mathcal{M} = (\mathcal{A}, X)$ is treated as private and y is public. If y is from a benign user Alice, it represents a particular task of interest. The prediction function $f_{\mathcal{M},y} : \mathcal{X}^p \to \mathcal{Y}$ is thus regarded as a particular model learned by $\mathcal{M}$ (Bob), driven by y, in order to provide assistance. Typically $f_{\mathcal{M},y}$ is also treated as private.

Definition 3 (Assisted Learning System). *An assisted learning system consists of a module M, a learning protocol, a prediction protocol, and the following two-stage procedure.*

- *In stage I ('learning protocol'), module $\mathcal{M}$ receives a user's query of a label vector $y \in \mathcal{Y}^n$ that is collated with the rows of X; a prediction function $f_{\mathcal{M},y}$ is produced and privately stored; the fitted value $f_{\mathcal{M},y}(X) = [f_{\mathcal{M},y}(x_1), \ldots, f_{\mathcal{M},y}(x_n)]^{\mathrm{T}}$ is sent to the user.*
- *In stage II ('prediction protocol'), module $\mathcal{M}$ receives a query of future predictor $\tilde{x}$; its corresponding prediction $\hat{y} = f_{\mathcal{M},y}(\tilde{x})$ is calculated and returned to the user.*

In the above Stage I, the fitted value, $f_{\mathcal{M},y}(X)$, returned from the service module (Bob) upon an inquiry of y, is supposed to inform the user module (Alice) of the training error so that Alice can take subsequent actions. Bob's actual predictive performance is reflected in the Stage II. The querying user in Stage II may or may not be the same user as in stage I.

Related work. In many recent MLaaS interfaces in industry, e.g., Google AI Cloud and Microsoft Azure, a service provider Bob helps a user Alice to construct learning models based on the labeled data uploaded by Alice. Bob holds the trained model private and provides a query interface for Alice to perform future predictions. This procedure is similar to our Stage II, except that Alice's data (including both labels and predictors) have to be held by Bob. In a decentralized scenario where the trustworthiness or learning capability of Bob is doubtful, Alice tends not to release private data. Assisted learning is more suitable in such scenarios.

A recent advancement in decentralized learning is Federated Learning [9]–[11], where the central server sends the current global model to a set of selected clients, and then each client updates the model parameter with local data and returns the updates back to the central server. The main idea of federated learning is to learn from machine learning models based on data that are distributed among participants (e.g., mobile devices) to avoid direct sharing of data, while all participants share the same model. In contrast, assisted learning is designed to allow diverse learning goals and models, and each participant can be either a user or service provider without the need of a central coordinator.

Another related homomorphic encryption framework is the Secure Multi-party Computation [12], [13]. The main idea is that any function can be computed securely, equivalently, it address that no players can learn anything more than its prescribed output. Several works [14], [15] under this framework studied machine learning on vertically partitioned data [16]–[18], which share certain similar feature with assisted learning. Secure Multi-party Computation naturally relies on the external service provider. This is different from assisted learning where each module is both service provider and learner. In addition, assisted learning is perhaps more suitable for large-scale network and complex dataset.

D. A Specific Learning Scenario: Iterative Assistance

Consider a situation where the goal is to improve the learning quality of a given learner, say Alice, by allowing its module, $\mathcal{M}_a = (\mathcal{A}_a, X_A)$ to exchange statistics with other m modules $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_m$. First, we note that for Alice to receive assistance from other modules, their data ($\mathcal{D}_1, D_2, ..., D_m$ and $\mathcal{D}_0 = X_A$) should be aligned or partially aligned (as defined in section II-A). We also consider a general setting where module k has features $\{X_i, i \in \mathcal{S}_j\}$, $j = 0, 1, \ldots, m$, and the feature sets $\mathcal{S}_k$ are overlapping, partially overlapping, or non-overlapping. Without any privacy constraint, it is natural to consider the following oracle performance as the limit of learning. Such limit has been widely adopted in classical statistical learning theory (see e.g. [19] and the references therein). Let ℓ denote some loss function (e.g. squared loss for regression).

Definition 4 (Oracle Performance). *The oracle performance of module $\mathcal{M}_a$ is defined by*

$$\min_{\mathcal{A}_j} \mathbb{E}\big\{\ell(y^*, \hat{A}_j(x^*))\big\}$$

where $\hat{A}_j$ is the trained $\mathcal{A}_j$ using all the pulled data $\bigcup_{j=0}^{m} \mathcal{D}_j$.

In other words, it is the optimal out-sample loss produced from the candidate methods and the pulled data of all modules (including Alice itself). The above quantity provides a theoretical limit or benchmark on what assisted learning can bring to Alice.

Suppose that Alice not only has a specific learning goal (labels), but also has private predictors and algorithm, how could Alice benefit from other modules/learners through the two stages of assisted learning? We address this by developing a specific user scenario of assisted learning. For Alice to receive assistance from other modules, their data should be at least partially collated. For brevity, we assume that the data of all the modules can be collated (defined in Subsection II-A) using public indices. Procedure 1 outlines a realization of assisted learning between Alice with m other modules. In the *training stage* (Stage I), at each round k, Alice first sends a query to each module $\mathcal{M}_j$ by transmitting its latest statistics $e_{j,k}$; upon receipt of the query, if module k agrees, it treats $e_{j,k}$ as labels and fit a model $\hat{\mathcal{A}}_{j,k}$ (based on the data aligned with such labels); module j then fits residual $\tilde{e}_{j,k}$ and sends it back to module 0. Alice processes the collected responses $\tilde{e}_{j,k}, \ldots$ ($j = 1, \ldots, m$), and initializes the $k+1$ round of communications. After the above procedure stops at an appropriate stopping time $k = K$, the training stage for Alice is suspended. In the *prediction stage* (Stage II), upon arrival of a new feature vector x^*, user 0 queries the prediction results $\hat{\mathcal{A}}_{j,k}(x^*)$ ($k = 1, 2, \ldots, K$) from module j, and combine them to form the final prediction $\tilde{y}^*$.

In Procedure 2, we consider regression methods and two modules for brevity. The key idea is to allow each module to only transmit fitted residuals to the other module, iterating until the learning loss is reasonably small. In particular, in the *training stage* (Stage I), Alice fits the data using her algorithm $\mathcal{A}_a$ and sends the fitted residuals e_1 to Bob. Then Bob treats them as labels and fits e_1 using his algorithm $\mathcal{A}_b$, and sends the fitted residuals $\tilde{e}_1$ back to Alice. Alice then initializes the second round of communication by treating $\tilde{e}_1$ as the new labels. Such a procedure repeats K times until the out-sample error (measured by, e.g., cross-validation) of Alice no longer decreases. In

the *prediction stage* (Stage II), for any new predictor, Alice queries the corresponding prediction from Bob using his trained models from the first stage, and forms the final prediction by suitably aggregating all the predictors.

Procedure 1 Assisted learning of Module 'Alice' with m other modules (general description)

Input: Module Alice and its initial label $y \in \mathbb{R}^n$, assisting modules $\mathcal{M}_1, \ldots, \mathcal{M}_m$, (optional) new predictors $\{x_i^*, i \in \mathcal{S}\}$
Initialisation: $e_{j,k} = y$ $(j = 1, \ldots, m)$, round $k = 1$
repeat
  Alice fits a supervised model using $(e_{j,k}, X_a)$ as labeled data and model $\mathcal{A}_a$.
  Alice records its fitted model $\tilde{\mathcal{A}}_{a,j,k}$ and calculates residual $r_{j,k}$.
  for j = 1 to m **do**
    Alice sends $r_{j,k}$ to $\mathcal{M}_j$.
    $\mathcal{M}_j$ fits a supervised model using $(r_{j,k}, X_j)$ as labeled data and model $\mathcal{A}_j$.
    $\mathcal{M}_j$ records its fitted model $\tilde{\mathcal{A}}_{j,k}$ and calculates residual $\tilde{e}_{j,k}$.
    $\mathcal{M}_j$ sends $\tilde{e}_{j,k}$ to Alice.
  end for
  Alice initializes the $k+1$ round by setting $e_{j,k+1} = \tilde{e}_{j,k}$
until Stop criterion satisfied

On arrival of a new data $\{x_i^*, i \in \mathcal{S}\}$, Alice queries prediction results produced by the recorded models $\tilde{y}_k = \hat{\mathcal{A}}_{j,k}(x_i^*, i \in \mathcal{S}_j) \in \mathbb{R}^n$, for $j = 1, \ldots, m$ and $k = 1, \ldots, K$.
Alice combines the above prediction results along with its own records to form a final prediction $\tilde{y}^*$.
Output: The *Assisted Learning* prediction $\tilde{y}^*$

Procedure 2 Assisted learning of Module 'Alice' ('a') using Module 'Bob' ('b') (one specific design)

Input: Module Alice and its initial label $y \in \mathbb{R}^n$, assisting module Bob, (optional) new predictors $\{x_i^*, i \in \mathcal{S}\}$
Initialisation: $e_k = y$, round $k = 1$
repeat
  Alice fits a supervised model using (e_k, X_a) as labeled data and model $\mathcal{A}_a$.
  Alice records its fitted model $\tilde{\mathcal{A}}_{a,k}$ and calculates residual r_k.
  Alice sends r_k to Bob.
  Bob fits a supervised model using (r_k, X_b) as labeled data and model $\mathcal{A}_b$.
  Bob records its fitted model $\tilde{\mathcal{A}}_{b,k}$ and calculates residual $\tilde{e}_k$.
  Bob sends $\tilde{e}_k$ to Alice.
  Alice initializes the $k+1$ round by setting $e_{k+1} = \tilde{e}_k$
until Stop criterion satisfied

On arrival of a new data $\{x_i^*, i \in \mathcal{S}\}$, Alice queries prediction results from Bob's local models: $\tilde{y}_{b,k}^* = \hat{\mathcal{A}}_{b,k}(x_i^*, i \in \mathcal{S}_b) \in \mathbb{R}^n$ for $k = 1, \ldots, K$.
Alice also calculates the prediction from her local models: $\tilde{y}_{a,k}^* = \hat{\mathcal{A}}_{a,k}(x_i^*, i \in \mathcal{S}_a) \in \mathbb{R}^n$ for $k = 1, \ldots, K$
Alice form a final prediction $\tilde{y}^* = \left(\sum_k \tilde{y}_{a,k}^*\right) + \left(\sum_k \tilde{y}_{b,k}^*\right)$.
Output: The *Assisted Learning* prediction $\tilde{y}^* \in \mathbb{R}^n$

Our theoretical analysis and numerical experiments will be based on this simplified Procedure 2, For theoretically analysis, it is straightforward to extend this result to the general Procedure 1. Our numerical experiments show that this result holds for a wide range of models including nonlinear models such decision tree method and ensemble methods as well. In the experimental section, an interesting observation is also presented, which is a tradeoff between communication complexity and learning performance that strikingly resembles the classical tradeoff between model complexity and learning performance. Besides, there are many interesting aspects to consider in order to facilitate more efficient assistance. Below we outline some interesting aspects that are worth future study.

- Multi-Armed Bandit & Online Learning. In deploying a practical assisted learning protocol, due to communication bandwidth, cost constraints, and computational overhead, Alice may only select a subset of m modules. It could be helpful to cast the module selection as a multi-armed bandit problem or expert learning problem.
- Adversarial Learning. In a network of modules, some of them may be malicious and propagate misleading results to Alice under assistance. Since our ultimate goal in Assisted Learning is to reduce bias and variance, it is crucial to detect adversarial modules.
- Design of efficient feedback information. In our preliminary Procedure 2, the final prediction is aggregated from each module at each communication round. While this achieves the oracle performance in many cases (see Section IV), it is not clear whether such design is the most efficient.
- Optimal stopping criterion. When to stop is a key ingredient part in assisted learning. On one hand, more communications typically to bring more information exchange and better fitting to the data. On the other hand, we have empirically observed that too frequent communications often bring overfitting so that the actual out-sample predictive performance of the module being assisted actually becomes worse.

Theorem 1. Suppose that Alice and Bob use linear regression models. Then for any label y, Alice will achieve the oracle performance for a sufficiently large number of communications k in Procedure 2.

Proof: In the appendix. It is straightforward to extend this result to the general assisted learning protocol as outlined in Procedure 1. The above result is applicable to linear models and additive regression models [20] on

a linear basis, e.g., spline, wavelet, or polynomial basis. Its proof is included in the supplementary material. The proof actually implies that the prediction loss decays exponentially with the number of communications. The result also indicates that if the true data generating model is $\mathbb{E}(y \mid x) = \beta_a^{\mathrm{T}} x_a + \beta_b^{\mathrm{T}} x_b$, where $x = [x_a, x_b] \in \mathbb{R}^p$ with a fixed p, then Alice achieves the optimal rate $O(n^{-1})$ of prediction loss as if Alice correctly specifies the true model. The results can be extended. For example, if x_a and x_b are independent, it can be proved that with one round of communications Alice can approach the oracle model with high probability for large data size n; and such an oracle loss approaches zero if $\mathbb{E}(y \mid x)$ can be written as $f_a(x_a) + f_b(x_b)$ for some functions f_a, f_b and if consistent nonparametric algorithms [21], [22] are used. Moreover, suppose that $\mathbb{E}(y \mid x)$ that cannot be written as $f_a(x_a)+f_b(x_b)$ but the interactive terms (such as $x_a \cdot x_b$ if both are scalars) involve categorical variables or continuous variables that can be well-approximated by quantizers. The Assisted Learning procedure could be modified so that Alice sends stratified dataset to Bob which involves only additive regression functions. An illustrating example is $\mathbb{E}(y \mid x) = \beta_a x_a + \beta_b x_b + \beta_{ab} x_{a,1} x_{b,1}$ where $x_{a,1} \in \{0, 1\}$, and Alice sends data $\{x_a : x_{a,1} = 0\}$ and $\{x_a : x_{a,1} = 1\}$ separately to Bob. In Section IV, we will show by experiments that the oracle performance can be well-approximated in general with a sufficiently number of communications.

E. Another Learning Scenario: Learning with Feedforward Neural Networks

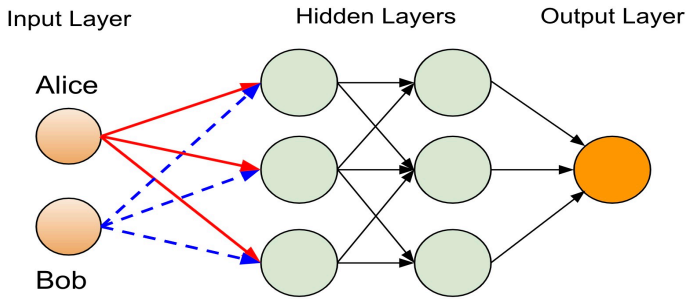


Fig. 2: Feedforward neural network via assisted learning. In each iteration, for the weights from input to hidden layers, Alice (Bob) can only update her (his) weights denoted by red solid lines (blue broken lines).

The last decade has witnessed the explosion of neural network techniques, and the extraordinary performances of deep neural network have made it the standard tool for several machine learning tasks. In this section, we present the learning protocol for feedfoward neural networks in the context of assisted learning.

The general setting will be the same as described in Section II-D. For simplicity, we consider the learning protocol of Alice and Bob with a three-layer feedforward neural network as depicted Fig. 2. Let $w_{a,k}$ (denoted by red solid lines) and $w_{b,k}$ (denoted by blue broken lines) be weights from input to hidden layers at the kth iteration for Alice and Bob respectively. Denote the rest weights in the neural network at the kth iteration by $\tilde{w}_k$. (Backpropagation is used to train the network) As summarized in Procedure 3, intuitively speaking, the weights for neural nets will be update in a coordinate-wise fashion, i.e., each module will and can only update those weights corresponding to its own data. This is because the data can never be shared between different modules in assisted learning for privacy concern. In the *training stage* (Stage I), at the kth iteration, Alice first calculates $w_{a,k}^{\mathrm{T}} X_A$, and inquiries Bob's $w_{b,k}^{\mathrm{T}} X_B$, then combine them to feed the neural network. If k is even, Alice will update her current weight $w_{a,k}$ to $w_{a,k+1}$ as well as $\tilde{w}_k$ to $\tilde{w}_{k+1}$. Bob will fix his weight for the next iteration, i.e. $w_{b,k+1} = w_{b,k}$. If k is odd, Alice will fix her weight for next iteration, i.e. $w_{a,k+1} = w_{a,k}$ and updates $\tilde{w}_k$ to $\tilde{w}_{k+1}$. Then she sends $\tilde{w}_k$ and other related information to Bob. Bob will use the information to update his current weight $w_{b,k}$ to $w_{b,k+1}$. Such a procedure repeats K times until the out-sample error (measured by, e.g., cross-validation) of Alice no longer decreases. In the *prediction stage* (Stage II), for any new predictor, Alice queries the corresponding $w_{b,K}^{\mathrm{T}} x_i^* (i \in \mathcal{S}_b)$ from Bob, and uses the trained neural network to get the assisted learning prediction.

In fact, based on the discussion from Section II-D, we notice that when the features held by Alice and Bob interact in a nonlinear fashion, then the iterative assistance scheme with nonlinear models typically will not lead to oracle scores. Even though this kind of situation rarely happens, the suboptimal learning performances are not consistent with the ideas of assisted learning. Fortunately, thanks to the strong representation and generalization power of neural networks, we believe the above proposed method can close this gap. We experimentally verify that the oracle score can be achieved by assisted learning with rare data distributions in Section IV-A3.

Procedure 3 Assisted learning of Module 'Alice' ('a') using Module 'Bob' ('b') for neural network

Input: Module Alice, its initial label $y \in \mathbb{R}^n$, initial weight $w_{a,1}$ (from input to hidden layers) and $\tilde{w}_1$ (the rest weights) for the neural network, assisting module Bob, (optional) new predictors $\{x_i^*, i \in \mathcal{S}\}$

Initialisation: round $k = 1$

repeat
  Alice calculates $w_{a,k}^{\mathrm{T}} X_A$ and receives Bob's $w_{b,k}^{\mathrm{T}} X_B$ to train the neural network
  if k is odd **then**
    Alice updates $w_{a,k}, \tilde{w}_k$ by using back-propagation to get $w_{a,k+1}, \tilde{w}_{k+1}$ respectively
    Bob sets $w_{b,k+1} \leftarrow w_{b,k}$
  else $\{k$ is even$\}$
    Alice sets $w_{a,k+1} \leftarrow w_{a,k}$ and updates $\tilde{w}_k$ by using back-propagation to get $\tilde{w}_{k+1}$
    Bob updates $w_{b,k}$ by using back-propagation to get $w_{b,k+1}$
  end if
  Alice initializes the $k+1$ round
until Stop criterion satisfied

On arrival of a new data $\{x_i^*, i \in \mathcal{S}\}$, Alice calculates $w_{a,K}^{\mathrm{T}} x_i^* (x_i^*, i \in \mathcal{S}_a)$.
Alice queries $w_{b,K}^{\mathrm{T}} x_i^*, (x_i^*, i \in \mathcal{S}_b)$ from Bob and combine them to feed the neural network to get the final prediction $\tilde{y}^*$.

Output: The *Assisted Learning* prediction $\tilde{y}^*$

III. Privacy at Module Level: 'Imitation Game'

A. Imitation Privacy

Concerns centering data privacy have led to more stringent regulations on the use of data in machine learning [23]. It has also raised various research interests in designing machine learning architectures that facilitate privacy as well as accuracy. A popular data-level privacy is the differential privacy [5]–[7] and its many variations including, local differential privacy [24], [25], concentrated differential privacy [26], Rényi differential privacy [27], and information-theoretic differential privacy [28], [29].

While data privacy is important in assisted learning, we also need a notion of privacy to protect the service provider is worried about potential leakage of his capability to generate algorithms during assisting others' learning. A competitor Alice may attempt to imitate the learning service provided by Bob, through consecutively querying or other side information. Such an imitation, if successful, will cause an undesirable leakage of Bob's black-box learning capability. In the rest of this section, we let $\mathcal{M} = (\mathcal{A}_b, X_B)$ denote the module of Bob and $f_{\mathcal{M},y}$ the

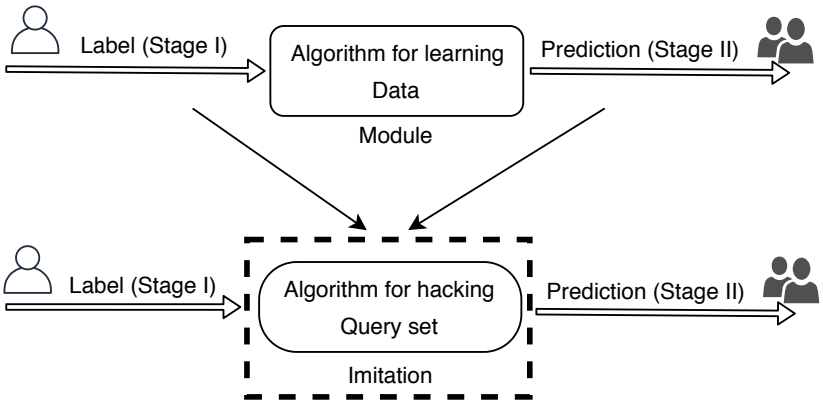


Fig. 3: Illustration of an Assisted Learning system (top) and its Imitation system (bottom).

learned model from label y. Let $\mathcal{I}$ denote any side information available to Alice, and $f_{\mathcal{I},y}$ the learned model of Alice using $\mathcal{I}$ and y. Mathematically, $\mathcal{I}$ could be treated as filtrations associated with an appropriate probability space.

Definition 5 (Imitation Privacy). *The imitation privacy for an imitation $\mathcal{I}$ and module M is*

$$\rho_{\mathcal{I},\mathcal{M}} = \mathbb{E}_{y \sim p_Y} \frac{\mathbb{E}_{\tilde{x} \sim p_X} |f_{\mathcal{I},y}(\tilde{x}) - f_{\mathcal{M},y}(\tilde{x})|^2}{\mathbb{E}_{\tilde{x} \sim p_X} |f_{\mathcal{M},y}(\tilde{x})|^2}, \tag{1}$$

where p_Y and p_X denote the distribution of query label y and (unobserved) future data $\tilde{x}$, respectively.

In the above definition, the denominator is to remove the unit of y. The definition is for regression scenarios, and it extensions for classification are left as future work. Without loss of generality, we assume $\mathbb{E}_{\tilde{x}\sim p_X}|f_{\mathcal{M},y}(\tilde{x})|^2 = 1$ in the rest of the paper.

Interpretations. For a given $\mathcal{M}$, smaller $\rho_{\mathcal{I},\mathcal{M}}$ means a closer imitation and less privacy. The minimal value $\rho_{\mathcal{I},\mathcal{M}} = 0$ is achieved at $f_{\mathcal{I},y} = f_{\mathcal{M},y}$ almost everywhere for every $y \in \mathcal{Y}^n$, meaning that Alice performs as well as Bob and there is '0' privacy for Bob. This can be clearly achieved when, for example, Alice holds both data and algorithm of Bob. The privacy value is typically greater than zero when Alice only holds side information such as a part of X_B, a transformation of X_B, or some other data that we will demonstrate in the sequel. On the other hand, the value of $\rho_{\mathcal{I},\mathcal{M}}$ is typically no larger than 1, since a trivial imitation $f_{\mathcal{I},y}(x) = 0$ for all x leads to $\rho_{\mathcal{I},\mathcal{M}} = 1$. As a result, it is expected that $\rho_{\mathcal{I},\mathcal{M}} \in [0, 1]$ and it is paramount to keep a large $\rho_{\mathcal{I},\mathcal{M}}$ for the benefit of Bob.

In the definition of $\rho_{\mathcal{I},\mathcal{M}}$, the closeness of $f_{\mathcal{I},y}$ and $f_{\mathcal{M},y}$ is evaluated on unobserved data (through $\mathbb{E}$). To enable easier computation, the privacy may be approximated by the training data $X = \{x_1, \ldots, x_n\}$ if x_i's are assumed to be i.i.d. generated. In other words, $\mathbb{E}$ may be replaced with $\mathbb{E}_n$, where $\mathbb{E}_n f(X) = n^{-1}\sum_{i=1}^n f(x_i)$ for any measurable function f.

The notion of imitation privacy may be extended in the following way. For two constants $\varepsilon, \delta \in [0, 1]$, the module $\mathcal{M}$ is said to be (ε, δ)-private with respect to $\mathcal{I}$ if with probability at most δ, $\rho_{\mathcal{I},\mathcal{M}} \leq \varepsilon$. The module $\mathcal{M}$ is said to be (ε, δ)-private with respect to a class of imitations $\mathfrak{I}$, if $\inf_{\mathcal{I}\in\mathfrak{I}} \rho_{\mathcal{I},\mathcal{M}} \leq \varepsilon$ with probability at most δ. The probability is due to possible randomizations of $\mathcal{I}$ or $\mathcal{M}$.

Example 1 (Algorithm Leakage). Consider a scenario where Bob's algorithm (Definition 1) is not available to Alice, but Bob's full data X_B and the fitted response $f_{\mathcal{M},y}(X_B)$ are available to Alice. Suppose that there is a small fraction of data that is mismatched or overly noisy, and Bob uses a robust learning algorithm to circumvent those outliers in X_B and learn an accurate model. Suppose that Alice holds a rudimentary algorithm that is sensitive to outliers. As a consequence of observing $f_{\mathcal{M},y}(X_B)$, Alice would be able to identify outliers as those with significant gaps between $f_{\mathcal{M},y}(X_B)$ and y. In this case, Bob's learning capability (in handling outliers) is implicitly leaked even if Bob's algorithm is not transmitted.

Example 2 (Data leakage). Consider a scenario where Bob's data X_B is not available to Alice, but his learning algorithm and fitted response $f_{\mathcal{M},y}(X_B)$ are available to Alice. A learning algorithm will demonstrate unique information regarding the dataset, e.g. column space revealed by linear regression and data structure implicitly shown from decision Tree. In some cases, Alice will be able to reverse-engineer some key statistics of Bob's data or even precise values of data, e.g. in Example 3 and 4 of the next section.

B. Imitation Privacy in Assisted Learning

In the context of an assisted learning system, how do we interpret and measure the privacy for the service module Bob? Suppose that a user module Alice has no prior information before contacting Bob in the Assisted Learning. The only way to gain information from Bob (or "hack the system") is through queries at either Stage I or State II (in Definition 3). Such information is quantified below.

Definition 6 (Query Set). *A query set $\mathcal{Q}$ from an Assisted Learning system consists of ordered quadruplets $\{(y_\ell, \hat{y}_\ell, \tilde{X}_\ell, \tilde{\mu}), \ell = 1, \ldots, k\}$, where*

1) y_ℓ is the ℓth query sent to the system in Stage I,

2) $\hat{y}_\ell$ is the fitted value returned by the system in Stage I,

3) $\tilde{X}_\ell \in \mathbb{R}^{n_\ell \times p}$ consists n_ℓ queries (by row) sent to the system during in II,

4) $\tilde{\mu}$ consists of predictions returned by the system in Stage II (that corresponds to $\tilde{X}_\ell$),

The above k is the number of Stage I queries, and $n_1, \ldots, n_k$ are the numbers of Stage II queries.

There are two components of a query set. The first component is concerned with the queries at Stage II that aim to hack $f_{\mathcal{M},y}(\cdot)$ *for a particular* y. The second component is to query Stage I in order to hack the internal functionality of $\mathcal{M}$ itself. The positive integer k in our context is interpreted as the communication complexity between modules. We will show by examples that a joint query to both Stage I and II is necessary to successfully imitate Bob.

If Alice is not a benign user, the above query set enables her to possess useful knowledge from the service module Bob. Simply speaking, Alice aims to provide assistance to other users whatever Bob could provide, as if

Alice had the algorithm and data that Bob privately holds. This is formulated by the following system in parallel to a regular assisted learning system.

Definition 7 (Imitation System). *An imitation $\mathcal{I}$ is a pair of query set $\mathcal{Q}$ and hacking algorithm $\mathcal{H}$ that maps $\mathcal{Q}$ and any label vector y to a prediction function $f_{\mathcal{I},y} : \mathbb{R}^p \to \mathbb{R}$ (which has the similar functionality as $f_{\mathcal{M},y}$).*

An imitation system $\mathcal{S}'$ consists of an imitation $\mathcal{I}$ from an assisted learning system $\mathcal{S}$, a learning protocol, a prediction protocol, and the two-stage procedure introduced in Definition 3, except that the prediction function, written as $f_{\mathcal{I},y}$, is produced from $\mathcal{I}$ instead of $\mathcal{M}$.

An illustration of the above concepts are included in Fig. 3. We now give some examples to concretely demonstrate the idea of Imitation Privacy. Technical details of these examples are included in the supplement.

Example 3 (Linear regression imitation privacy). Suppose that in an imitation, the number of Stage I queries is k_1, and the numbers of Stage II queries are $n_1 = \cdots = n_k = k_2$. Suppose that Bob employs a linear regression model and Alice knows about it. By querying $k_1 \geq \min\{p, n-p\}$ random label vectors y_ℓ ($\ell = 1, \ldots, k_1$) from Stage I, Alice can obtain the column space $\text{span}(X_B)$ of Bob's data X_B with probability one. Additionally, if Alice also knows about the true covariance matrix of Bob's features, Alice is able to develop an imitation system with $\rho_{\mathcal{I},\mathcal{M}} = o(n^{-1})$, where n is the data size (or the number of rows in X_B). Note that Bob's data are never transmitted.

Solution. Suppose that in an imitation, the number of Stage I queries is k_1. Suppose that Bob employs a linear regression model and Alice knows that Bob is using a linear regression. By querying $k_1 \geq \min\{p, n-p\}$ random label vectors y_ℓ ($\ell = 1, \ldots, k_1$) from Stage I, Alice can obtain the column space $\text{span}(X_B)$ of Bob's data X_B with probability one. This is because in each fitting process, Bob will project the random query y_i onto $span(X_B)$, i.e. $y_i \mapsto P_{X_B} y_i$. Hence, with $k_1 = p$ fitted values in the form of $P_{X_B} y_i$, $i = 1, 2, ...k_1$, Alice is able to uniquely identify the column space of X_B with probability 1. On the other hand, with $n - p$ queries, Alice will identify the orthogonal space of $span(X_B)$, which further implies $span(X_B)$.

However, Alice cannot obtain the Bob's data X_B exactly without further side-information. One such side information is the true covariance matrix of Bob. Alice is able to develop an imitation system with $\rho_{\mathcal{I},\mathcal{M}} = o_p(1)$ as Bob's data size $n \to \infty$. To see this, suppose without loss of generality that the underlying covariance of X_B is an identity matrix. Let Alice arbitrarily pick up a matrix $\tilde{X} \in \mathbb{R}^{n \times p}$ whose column space is $span(X_B)$. We only need to find such a Q that $\tilde{X} = X_B Q$. For each label $Y_t, t = 1, 2, ..., p$ sent in Stage I, Alice can calculate the empirical covariance between Y and X_B, say $K_t \in \mathbb{R}^{p \times 1}$. By the law of large numbers, $K_t \to_p cov(QX_B, Y_t) = Qcov(X_B, Y_t)$ as $n \to \infty$. If each query Y_t is in the form: $Y_t = X_B \beta_t + \eta_t$, with some fixed $\beta_t \in \mathbb{R}^{p \times 1}$, then Alice could solve Q by letting

$$\begin{bmatrix} K_1 & K_2 & \ldots & K_p \end{bmatrix} = Q \begin{bmatrix} \beta_1 & \beta_2 & \ldots & \beta_p \end{bmatrix}$$

As long as $\boldsymbol{\beta}$ is linearly independent, Alice obtains a unique $\hat{Q}_n$ that converges in probability to the true Q. Consequently, Alice would use $\tilde{X}\hat{Q}_n^{-1}$ as if it were X_B to provide assistance, with an $o_p(1)$ imitation privacy. □

Example 4 (Decision tree imitation privacy). Suppose that Bob uses a decision tree with width 2 and depth at least p, with p being the number of Bob's features. Then with $k_1 \geq n$ and $k_2 = \infty$, there exists an imitation such that $\rho_{\mathcal{I},\mathcal{M}} = 0$ with high probability.

Solution. Suppose that Bob uses a decision tree with width 2 and depth at least p, with p being the number of Bob's features. Then with $k_1 \geq n$ and $k_2 = \infty$, there exists an imitation such that $\rho_{\mathcal{I},\mathcal{M}} = o_p(1)$ as $n \to \infty$. In fact, in the ith Stage I query, Alice sends label y_i such that its ith entry is sufficiently large and all other entries are zero ($i = 1, \ldots, n$) to identify the structure of Bob's data. From the infinite Stage II queries corresponding to the ith Stage I query, Alice is able to reconstruct the tree built by Bob, which puts a mass at the finest neighborhood of x_i. Finally, Alice is able to reconstruct Bob's data up to a precision that goes to 0 as n grows. Therefore, with $k_1 \geq n$ and $k_2 = \infty$, by using the strategy described above, Alice can create an imitation system such that $\rho_{\mathcal{I},\mathcal{M}} = o_p(1)$.

In fact, there exist multiple ways to obtain the data structure of Bob. Next, we demonstrate another way that does not even need Stage II queries. Consider a one-dimensional case where Bob's data is $X_B^{\mathrm{T}} = [7, 1, 10, 5, 18, 9]$, and a decision tree with width 2 is employed. We use x_j to denote the jth entry in X_B^{T}. Alice can obtain the structure

of Bob's data by sending queries e_i, for $i = 1, 2, ..., 6$ to Bob in Stage I only, where e_i is the standard basis for $\mathbb{R}^6$, and observing the fitted values o_i for $i = 1, 2, ..., 6$.

Input	$\xrightarrow{\mathcal{M}}$	Fitted Value
e_1 = [1,0,0,0,0,0]		o_1 = [$\frac{1}{3}$,$\frac{1}{3}$,0,$\frac{1}{3}$,0,0]
e_2 = [0,1,0,0,0,0]		o_2 =[0,0,0,0,0,0]
e_3 = [0,0,1,0,0,0]		o_3 =[0,0,$\frac{1}{2}$,0,$\frac{1}{2}$,0]
e_4 = [0,0,0,1,0,0]		o_4 =[0,$\frac{1}{2}$,0,$\frac{1}{2}$,0,0]
e_5 = [0,0,0,0,1,0]		o_5 =[0,0,0,0,0,0]
e_6 = [0,0,0,0,0,1]		o_6 =[0,0,$\frac{1}{3}$,0,$\frac{1}{3}$,$\frac{1}{3}$]

From o_2 and o_5, Alice knows that x_2, x_5 are beginning/ending points, and without loss of generality, we assume $x_2 < x_5$. From o_4 and o_1, Alice knows x_4 must lie between x_1 and x_2. Similarly, the order of x_6, x_3, x_5 can be inferred from o_3, o_6. Therefore, Alice has successfully recovered the structure of Bob's data, i.e., $[x_2, x_4, x_1, x_6, x_3, x_5] = [1, 5, 7, 9, 10, 18]$. Note that without the information from Stage II, Alice can never know the exact value or even ranges of Bob's data. In fact, it is straightforward to apply such kind of strategy on complex dataset.

□

Example 5 (Restricted outcome imitation privacy). Suppose that y is generated from $y = f(x) + \eta$, where $\eta \sim \mathcal{N}(0, \sigma^2)$ and f is randomly generated from a compact space $\mathcal{F}$ with a suitable probability measure F and metric $L_2(P_X)$. The distribution of y conditional on Bob's data X, $p_{Y|X}$, is Gaussian with mean $\int_{\mathcal{F}} f(X)dF$ and variance σ^2. Then there exists an imitation $\mathcal{I}$ with $k_1 = \exp\{H_{d,\varepsilon}(\mathcal{F})\}$ and $k_2 = \infty$, such that $\rho_{\mathcal{I},\mathcal{M}} \leq \varepsilon$ with high probability, where $H_{d,\varepsilon}(\mathcal{F})$ denotes the Kolmogorov ε-entropy of $\mathcal{F}$.

Solution. Suppose that y is generated from $y = f(x) + \eta$, where $\eta \sim \mathcal{N}(0, \sigma^2)$ and f is randomly generated from a compact space $\mathcal{F}$ with a suitable probability measure F and metric $L_2(P_X)$. The marginal distribution of y conditional on Bob's data X, $p_{Y|X}$, is thus Gaussian with mean $\int_{\mathcal{F}} f(X)dF$ and variance σ^2. Then there exists an imitation $\mathcal{I}$ with $k_1 = \exp\{H_{d,\varepsilon}(\mathcal{F})\}$ and $k_2 = \infty$, such that $\rho_{\mathcal{I},\mathcal{M}} \leq \varepsilon$ with high probability, where $H_{d,\varepsilon}(\mathcal{F})$ denotes the Kolmogorov ε-entropy of $\mathcal{F}$, namely the logarithm of the smallest number of ε-covers of $\mathcal{F}$. In fact, Alice may be able to develop the following imitation system.

Let $f_1, \ldots, f_{k_1}$ with $k_1 = \exp\{H_{d,\varepsilon}(\mathcal{F})\}$ denote the ε-quantizations of the function space $\mathcal{F}$. Suppose that k_1 queries are constructed in such a way that y_j is generated from f_j, namely $y_j = f_j(x) + \eta_j$, $j = 1, \ldots, k_1$. For any future query sent to Alice, say $y_* = f_*(x) + \eta_*$, Alice can search from the dictionary of y_j, $j = 1, \ldots, k_1$, and find the j that minimizes $\|y_j - y_*\|$. Since $n^{-1}\|y_j - y_*\|^2 = 2\sigma^2 + \|f_j - f_*\|_{L_2(P_X)} + o_p(1)$ as $n \to \infty$ (assuming independent noises), Alice would obtain such j that f_j is ε-away from f_*. Consequently, when Alice uses the Stage II queries corresponding to y_j to assist others, she obtains an imitation privacy of $\rho_{\mathcal{I},\mathcal{M}} \leq \varepsilon$ with high probability (for large n). □

Related work. A related concept is Optimal Experimental Design or Active Learning [30], [31], where the focus is to study economical collection of labeled data to train a learning algorithm with comparable accuracy. Model Extraction [32], [33], the process of reconstructing machine learning models through prediction APIs, is similar to the hacking process in Stage II. In model extraction, the goal is to extract specific model trained on given label. While in assisted learning, we aim to acquire the ability of mimicking the functionality of module, which shall hold for arbitrary label.

Knowledge distillation [34], [35] utilizes information from complex model (teacher network) to train a smaller one (student network) with comparable accuracy. The student network will be trained on 'soft target', which is teacher network's output. This is similar to the idea that 'key statistics' transmitted so as to improving learning performance in assisted learning.

Another closely related concept is differential privacy. The main goal of differential privacy is to secure the privacy of data. In the context of imitation privacy, the focus is the to secure the privacy of both the data and learning model. Below we give two examples to demonstrate that differential privacy and imitation privacy do not imply each other.

Example 6 (Ensured differential privacy and breached imitation privacy). Suppose that Bob's data X_B contains n i.i.d. observations of a random variable supported on $[-b, b]$. Bob can apply Laplacian mechanism to his data X_B to get α-locally deferentially private data $\tilde{X}_B$, and then releases $\tilde{X}_B$ to Alice.

However, the above mechanism typically does not admit a non-vanishing imitation privacy (Definition 5, for any label distribution p_Y). For example, suppose that Bob uses a linear regression model, then Alice can create the following imitation system with a vanishing imitation privacy. For any queried y, Alice calculates $\hat{\beta}_a = (\tilde{X}_B^{\mathrm{T}}\tilde{X}_B - \tau^2 I)^{-1}\tilde{X}_B^{\mathrm{T}}y = (\tilde{X}_B^{\mathrm{T}}\tilde{X}_B - \tau^2 I/n)^{-1}(\tilde{X}_B^{\mathrm{T}}y/n)$, and uses $f_{\mathcal{M},y} : x \mapsto \hat{\beta}_a^{\mathrm{T}}x$ for prediction, where $\tau^2 = 8b^2/\alpha^2$ is the variance of the Laplacian noise that could be estimated from Stage II if not known to Alice. By the law of large numbers, the above $\hat{\beta}_a$ converges in probability to the same limit as Bob's estimator $\hat{\beta}_b = (X_B^{\mathrm{T}}X_B)^{-1}X_B^{\mathrm{T}}y$. This implies a vanishing imitation privacy as the data size n becomes large.

Example 7 (Ensured imitation privacy and breached differential privacy). Suppose that a module Bob is equipped with a linear regression algorithm. Suppose that one predictor/feature is released to the public, then his dataset will not be differentially private at any privacy level. However, such direct release of partial data will only decrease the imitation privacy by a small amount. Alice still can not estimate the functionality of Bob with arbitrary accuracy.

IV. Experimental Study

We provide numerical demonstrations of the proposed method in Section II-D, II-E. For synthetic data, blue line represents the mean of training errors from 20 replicates, and the red line stands for the mean of test errors from 20 replicates. The oracle performance (denoted by black dashed line) is the testing error obtained by the model trained on the pulled data. In each replicate, we use a training dataset with size 500, and a testing data with size 10000. We chose a testing size much larger than training size in order to produce a fair comparison of out-sample predictive performance (see [8], [19] for a detailed discussion). For real data, the red line indicates mean of the testing error on a testing dataset which is 30% of whole data, resampled 20 times. In addition, the oracle performance (denoted by black dashed line) is the testing error obtained by the model that is previously trained on the pulled data. Also, the shaded regions describe the corresponding -1/+1 standard errors. Besides, for all nonlinear learning algorithms, tuning-parameters are finely tuned.

A. Synthetic Data

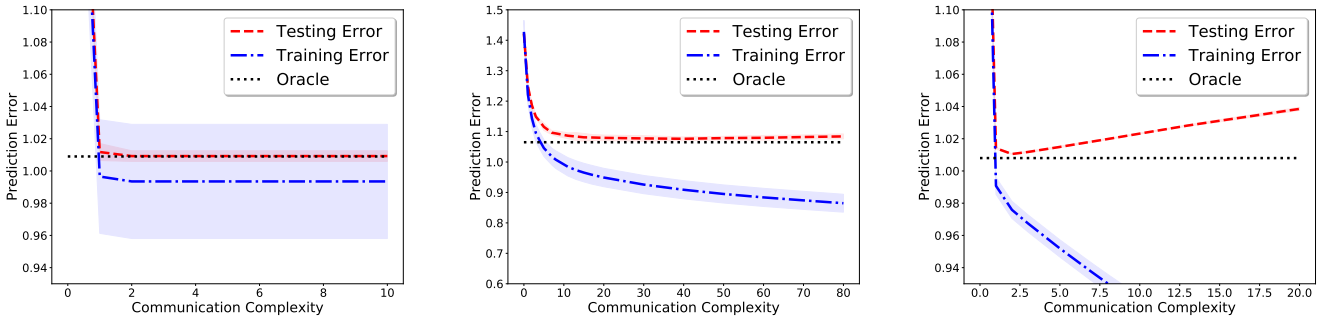
1) Synthetic Data with linear underlying pattern: We first consider the case where the true data generating function is linear. Let $\mathbf{x}_j = [x_{j1}, x_{j2}, \ldots, x_{j6}]$, where $x_{ji} \overset{IID}{\sim} \mathcal{N}(0,1)$. The data generating model is $y = X\beta + \varepsilon$, where $\beta \sim \mathcal{N}(\mathbf{0}, I_6)$, and $\varepsilon \sim \mathcal{N}(0,1)$. Module A holds data $X_A = [x_1, x_2, x_3]$ and Module B holds data $X_B = [x_4, x_5, x_6]$. The experiments are independently replicated 20 times. Each time a training size of 500 and a testing size of 100000 are used.

Figures below show the prediction performances of module A with 3 different learning algorithms. In Fig. 4a, with linear regression model, we observe that the error terms converge in two iterations and exactly match the oracle score, which verifies the statement of 'convergence in one round with high probability for independent data' in Theorem 1. Fig. 4b is the prediction performance of using decision tree (regression). The error term decreases to the oracle score with 25 rounds of communication. Gradient boosting algorithm is used in Fig. 4c. Note that the testing error first decreases to the oracle score and then begin to increase. Interestingly, this phenomenon strikingly resembles the classical tradeoff between overfitting and underfitting due to model complexity. In our case the communication complexity is the counterpart of model complexity.

2) Synthetic Data with non-linear underlying pattern: We next test on the data generated by Friedman1 benchmark [36], [37], i.e.,

$$f(\mathbf{x}) = 10\sin(\pi x_1 x_2) + 20(x_3 - \frac{1}{2})^2 + 10x_4 + 5x_5 + \varepsilon, \tag{2}$$

where $\varepsilon \sim \mathcal{N}(0,1)$ and $x_i \overset{IID}{\sim} Unif(0,1)$ for $i = 1, 2, ..., 5$. Module A has the first 2 features $X_A = [x_1, x_2]$ and Module B has the rest 3 features $X_B = [x_3, x_4, x_5]$. Fig. 5a, 5b, below show the prediction performances of module A with linear regression and additive model with spline basis. We observe that both methods attain their corresponding oracle scores, and the error terms converge very fast, which is expected from Theorem 1. Fig. 5c, 5d, 5e are the prediction performances of module A with decision tree (regression), random forest, and gradient boosting. All the error terms achieve their corresponding oracles, and similarly, we observe the overfitting issue with respect to the communication complexity.



(a) Model: Orindary linear regression (b) Model: Decision Tree (c) Model: Gradient Boosting

Fig. 4: Prediction performances for Module A (as measured by RMSE) on synthetic linear data with multiple learning algorithms. Each line is the mean of 20 replicates and the shaded regions describe the corresponding +1/-1 standard errors. In (a), with linear regression model being employed, the testing error converges in two iterations and exactly matches the oracle score. In (b), with decision tree (regression) being used, the testing error decreases to the oracle score around 25 iterations. In (c), with gradient boosting model, the testing error first decreases to the oracle score and then begin to increase. (Overfitting with respect to the communication complexity.)

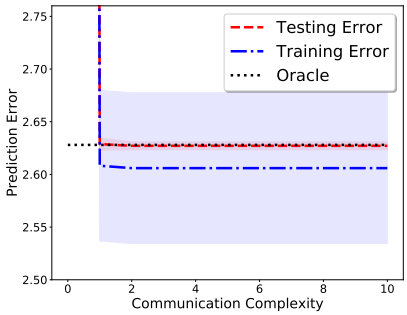
In fact, from the data generating function eq. (2), we notice that the interaction effect is only between covariates x_1 and x_2. The effects contributed by other features are independent. Therefore, it is reasonable for module A with features $[x_1, x_2, x_3]$ to achieve the oracle score in assisted learning, since the interaction term can be well learned by the machine learning model. Next, we consider the case where module A holds features $[x_1, x_3, x_4]$ and module B holds features $[x_2, x_5]$. Fig. 5f demonstrates the prediction performances of module A using gradient boosting. There is a gap between the oracle score and the assisted learning result. Such gap is due to the fact that interaction terms x_1 and x_2 are now fitted separately. However, compared to the result without using assisted learning assisted learning, it does bring significant improvement to module A. Indeed, this gap can be closed by neural network protocol in assisted learning, and we shall numerically address this issue in the upcoming section.

3) Assisted Learning for Neural Network: In addition, we use two-layer neural network, with 4 nodes in the hidden layer and Relu function, to illustrate the proposed method. The training size is 500 and the testing size is 10000 for each replicate. Full batch is used to train the network each time. We test on the same two datasets used in the previous section. Module A holds features $[x_1, x_3, x_4]$ and module B will hold the rest features corresponding to each dataset. We follow the learning protocol described in Section II-E. Fig. 6a, 6b show the prediction performances of module A on linear data and Friedman 1 respectively. In both cases, assisted learning eventually approaches the performance of vanilla neural network (oracle). Note that the features held by A are not complete for interaction term, i.e. x_1 and x_2 are separate for the Friedman1 data. Yet, the oracle performance is achieved by assisted learning and therefore we solve the problem raised in the previous section.

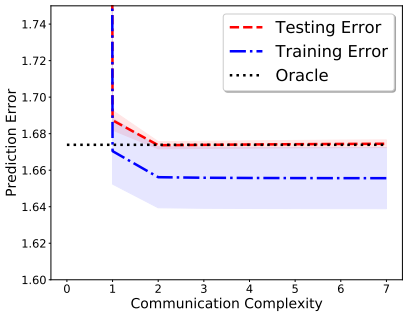
B. Real Data Study

We demonstrate our approach using the *Superconductor Data* [38] that consists of 21263 entries and 81 features. The learning task is to use chemical characteristics to predict the superconducting critical temperatures. The features is partitioned into two sets, 40 features held by module A and the other 41 features held by B. We consider two settings where one module uses gradient boosting and the other one uses linear regression.

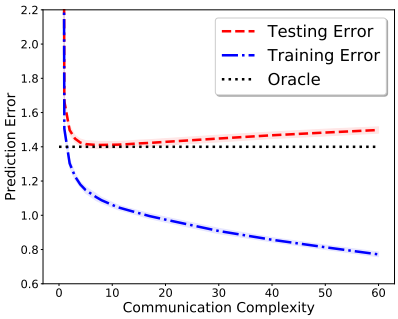
The results as depicted in Fig. 7a, 7b show that both modules can approximately achieve their corresponding oracle performances. The prediction performance for gradient boosting is much better than Linear Regression. In terms of convergence rate, gradient boosting converges faster than linear regression. In Fig. 7b, we obverse the 'over-fitting' issue with respect to communication complexity.



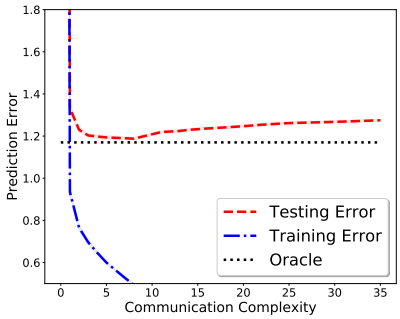
(a) Data: Friedman1
Model: Ordinary linear regression



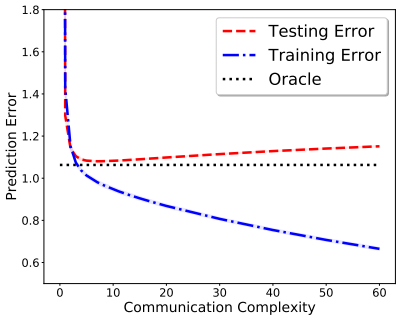
(b) Data: Friedman1
Model: Additive model with spline basis



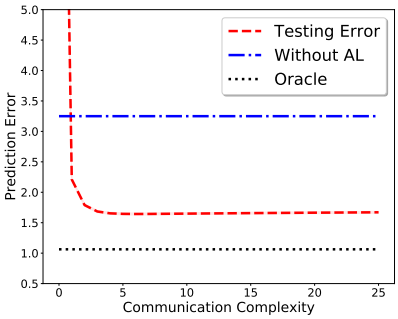
(c) Data: Friedman1
Model: Decision tree



(d) Data: Friedman1
Model: Random forest



(e) Data: Friedman1
Model: Gradient boosting



(f) Data: Friedman1
Model: Gradient boosting

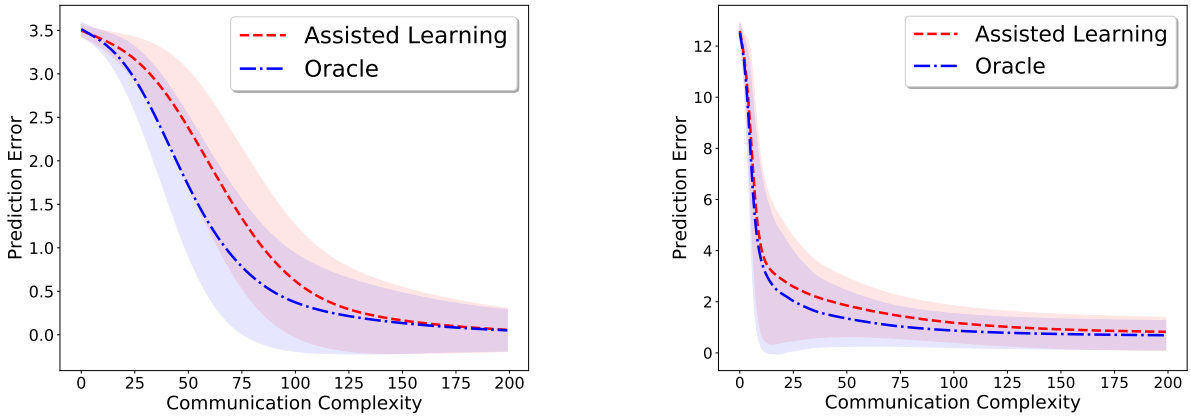
Fig. 5: Prediction performances for Module A (as measured by RMSE) on Friedman 1 dataset with multiple learning algorithms. Each line is the mean of 20 replicates and the shaded regions describe the corresponding +1/-1 standard errors. In (a) and (b), with least square based method, testing errors rapidly converges to the corresponding oracle scores. In (c), (d) and (e), with nonlinear methods, we observe the overfitting issue with respect to the communication complexity. In (f), the distribution of features is different from the previous cases, which is a quite rare scenario. There is a gap between the assisted learning and the oracle performance. Yet, assisted learning does bring significant improvement compared with the result without it.

C. The interactions between Federated Learning and Assisted Learning

In this section, we demonstrate the interactions between federated learning and assisted learning. We will first see how these two frameworks benefit from each other. Suppose that the whole feature space of interests is $[x_1, x_2, ..., x_p] \in \mathbb{R}^p$ and Alice has n_1 data entries, each with features $[x_1, x_2, x_3]$. Consider the following 2 types of learners: 1. C-type learners, who hold the whole features $[x_1, x_2, ..., x_p]$ but with limited number of data entries; 2. B-type learners, who holds partial features, e.g. $[x_3, x_5, x_6]$, and the data entries can be aligned (fully or partially) with Alice's data. C-types learners are the typical participants in federated learning and B-type learners are commonly seen in assisted learning. Now, Alice wants to learn a supervised relationship with her data, what is a good strategy for her in the presence of other 2-types of learners?

First, with Alice's own data only, the learned model $\hat{f}_A$ will certainly be biased. So it is beneficial for Alice to acquire type-B learners' information to modify the biased model, which can be achieved via assisted learning framework. If Alice manages to obtain a 'correct' model $\tilde{f}_A$, how can she further improve her predictive performance? A possible way is for Alice to communicate with C-type learners by averaging their models, i.e. $\bar{f}_A \leftarrow w_a \tilde{f}_A + \sum_i w_{c,i} \hat{f}_{C,i}$, to gain a better prediction performance The above process can be implemented in the framework of federated learning.

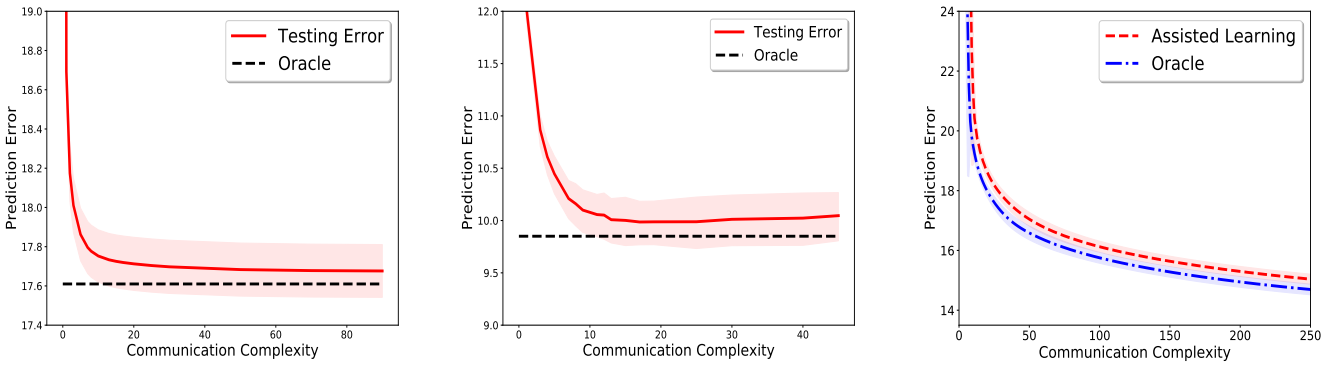
In detail, for experimental study, there are in total 12 features $X = [x_1, x_2, ..., x_{12}] \in \mathbb{R}^{12}$ with $x_i \overset{IID}{\sim} \mathcal{N}(0, 1)$



(a) Prediciton performances of module A with two-layer neural network on linear data.

(b) Prediciton performances of module A with two-layer neural network on Friedman 1.

Fig. 6: Prediction performances (as measured by RMSE) of module A with two-layer neural work on two different datasets. Oracle means the result obtained from vanilla two-layer neural network. Each line is the mean of 100 replicates and the shaded regions describe +1/-1 standard errors.



(a) Data: Superconductor Data
Model: linear regression

(b) Data: Superconductor Data
Model: gradient boosting

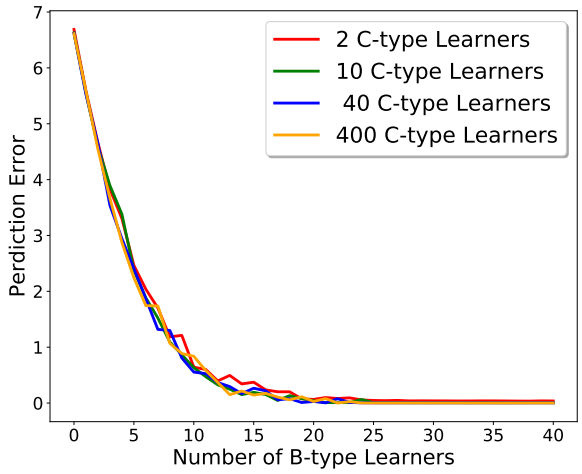
(c) Data: Superconductor Data
Model: Two-layer Neural Network

Fig. 7: Prediction performances for Module A (as measured by RMSE) on Superconductor Data with two learning algorithms. Each line is the mean of 20 replicates and the shaded regions describe +1/-1 standard errors.

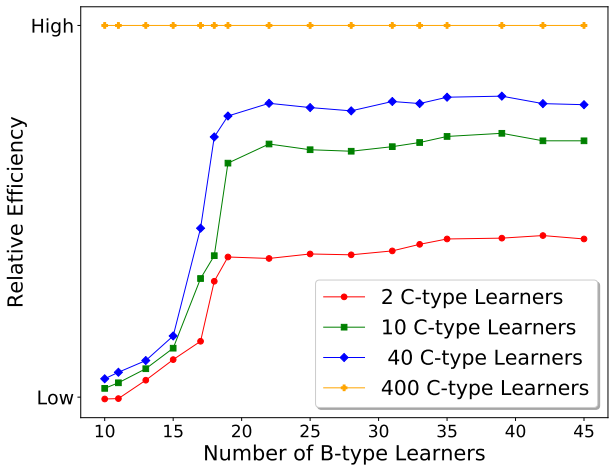
for $i = 1, 2, ..., 12$, and the data is generated from $y = X\beta + \varepsilon$, with $\varepsilon \sim \mathcal{N}(0, 1)$ and $\beta \sim \mathcal{N}(\mathbf{0}, I_{12})$. Alice holds $[x_1, x_2, x_3]$ and has 100 data entries, and each B-type learner will hold 3 randomly selected features. All learners will employ linear regression as machine learning models. Alice follows Procedure 1 to communicate with B-type learners, and will communicate each of them only once ($K = 1$ in Procedure 1). Each C-type learner will have the whole 12 features and 25 data entries. Alice will follow the *Fedavg* algorithm and serve as the central service in federated learning to communicate with C-type learners. There are 4 lines in Fig. 8a, with each single line representing the mean of 100 replicates. For convenience, we denote C_1 to be the learning performance with 2 C-type Learners (red line in the figure (a)), and similarly for C_2, C_3, C_4.

In Fig. 8a, we see that as the number of B-type learners increases, the out sample prediction performances (on 10000 data points, as measured by RMSE) significantly reduce and eventually go to 0. This shows the significant benefit brought by assisted learning. To see the goodness of federated learning, we calculate the following ratios: $\frac{C4}{C1}, \frac{C4}{C2}, \frac{C4}{C3}$, and $\frac{C4}{C4}$, to use them as the statistical efficiency measurement (High = 1, Low = 0). In Fig. 8b, we

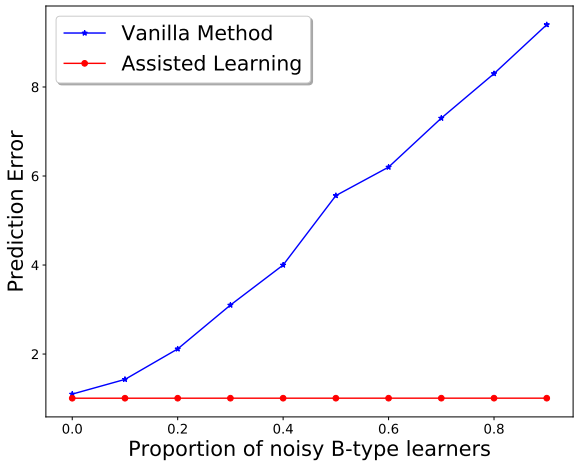
observe that when B-typer learners increase from 10 to 20, the ratios $\frac{C4}{C1}$, $\frac{C4}{C2}$ and $\frac{C4}{C3}$ increase due to the reduction of bias term in mean square error (MSE). After the number of B-type learners exceeding 25, the bias term almost goes to 0, and we see the benefit on efficiency brought by federated learning.



(a) Prediciton performances of Alice under multiple scenario as measured by RMSE



(b) Relative efficiency (High =1, Low= 0)



(c) Predicition performances of Alice with noisy B-type learners as measured by RMSE

Fig. 8: The interactions between Federated Learning and Assisted Learning. In (a), each line corresponds to a set of C-type learners. For all kinds of C-type learners, as the number of B-type learners increase, the prediction errors decreases to 0 eventually. In (b), the relative efficiency is calculated as the ratio of results in (a) to '400 C-type Learners' respectively. The red dotted line is the ratio of '400 C-type Learners' to '2 C-type Learners', the green square line is the ratio of '400 C-type Learners' to '10 C-type Learners', the blue diamond line is the ratio of '400 C-type Learners' to '40 C-type Learners', and the yellow crossed line is the ratio of '400 C-type Learners' to '400 C-type Learners'. In (c), with vanilla method, the prediction error quickly goes up as the proportion of noisy B-type learners increase.

The above experiments show that these two frameworks can help each other to achieve a better learning performance. Next, we will demonstrate a unique advantage of assisted learning. As we discussed before, the choice of machine learning model is arbitrary in assisted learning. Consider the scenario where there are only

B-type learners and a proportion of B-type learners have very noisy data. If they are all restricted to use the same model, e.g., ordinary linear regression, then the out-sample prediction performance can be very bad. However, if they follow the assisted learning protocol, i.e. learners with noisy data opt to robust learning technique, then the out-sample prediction performances will be very promising.

For the experimental study, we have in total 20 B-type learners, and learner B_j for some $j = 1, 2, \ldots, [20\rho]$ have really noisy data. Here ρ is the proportion of clients whose data contain outliers and larger noise. For vanilla method, all the learners will use the linear regression model. For assisted learning, those B-type learners with noisy data will use robust regression technique, and those normal B-type learners will use linear regression. Fig. 8c demonstrates the prediction performances of Alice for two proposed methods with different proportions of noisy B-type learners. With vanilla method, the RMSE significantly goes up as the portion of noisy B-type learners increases. For assisted learning, the RMSE remains around 1 in all cases.

V. Conclusion

The interactions between multiple learners in privacy-aware scenarios pose new challenges that cannot be well addressed by classical statistical learning with a single learning objective and algorithmic procedure. In this work, we propose the notion of Assisted Learning, where the key idea is to treat predictors as private and labels as public, and learners hold private predictors as well as private algorithms provide learning services for other learners.On the other hand, most of the existing literature on privacy focuses on protecting users' data, there is also a growing demand for protecting the learners who manage data. A new notion of privacy, imitation privacy, is proposed to quantify the privacy leakage of a learner who provides assistance. This privacy enables a unified measurement for both data as well as private model being used.

REFERENCES

[1] M. M. Alabbadi, "Mobile learning (mlearning) based on cloud computing: mlearning as a service (mlaas)," in *Conference on Mobile Ubiquitous Computing*, 2011.

[2] M. Ribeiro, K. Grolinger, and M. A. Capretz, "Mlaas: Machine learning as a service," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2015, pp. 896–902.

[3] Y. Yang, "Regression with multiple candidate models: selecting or mixing?" *Statistica Sinica*, pp. 783–809, 2003.

[4] J. Ding, V. Tarokh, and Y. Yang, "Model selection techniques: An overview," *IEEE Signal Process. Mag.*, vol. 35, no. 6, pp. 16–34, 2018.

[5] C. Dwork and K. Nissim, "Privacy-preserving datamining on vertically partitioned databases," in *Annual International Cryptology Conference*. Springer, 2004, pp. 528–544.

[6] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2006, pp. 486–503.

[7] C. Dwork, "Differential privacy," *Encyclopedia of Cryptography and Security*, pp. 338–340, 2011.

[8] G. Claeskens, N. L. Hjort *et al.*, "Model selection and model averaging," *Cambridge Books*, 2008.

[9] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. ACM, 2015, pp. 1310–1321.

[10] J. Konecny, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, "Federated learning: strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[11] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.

[12] A. C.-C. Yao, "How to generate and exchange secrets," in *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. IEEE, 1986, pp. 162–167.

[13] O. Goldreich, "Secure multi-party computation," *Manuscript. Preliminary version*, vol. 78, 1998.

[14] A. Gascón, P. Schoppmann, B. Balle, M. Raykova, J. Doerner, S. Zahur, and D. Evans, "Secure linear regression on vertically partitioned datasets." *IACR Cryptology ePrint Archive*, vol. 2016, p. 892, 2016.

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

[16] J. Vaidya and C. Clifton, "Privacy-preserving k-means clustering over vertically partitioned data," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003, pp. 206–215.

[17] ——, "Privacy preserving naive bayes classifier for vertically partitioned data," in *Proceedings of the 2004 SIAM International Conference on Data Mining*. SIAM, 2004, pp. 522–526.

[18] ——, "Privacy-preserving decision trees over vertically partitioned data," in *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 2005, pp. 139–152.

[19] J. Ding, V. Tarokh, and Y. Yang, "Model selection techniques: An overview," *IEEE Signal Process. Mag.*, vol. 35, no. 6, pp. 16–34, 2018.

[20] C. J. Stone, "Additive regression and other nonparametric models," *Ann. Stat.*, pp. 689–705, 1985.

[21] ——, "Consistent nonparametric regression," *Ann. Stat.*, pp. 595–620, 1977.

[22] G. Biau, "Analysis of a random forests model," *Journal of Machine Learning Research*, vol. 13, no. Apr, pp. 1063–1095, 2012.

[23] P. Voigt and A. Von dem Bussche, "The eu general data protection regulation (gdpr)," *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 2017.

[24] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *IEEE Symp. FOCS*, 2013, pp. 429–438.

[25] S. Xiong, A. D. Sarwate, and N. B. Mandayam, "Randomized requantization with local differential privacy," in *IEEE Proc. ICASSP*, 2016, pp. 2189–2193.

[26] C. Dwork and G. N. Rothblum, "Concentrated differential privacy," *arXiv preprint arXiv:1603.01887*, 2016.

[27] I. Mironov, "Rényi differential privacy," in *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE, 2017, pp. 263–275.

[28] F. du Pin Calmon and N. Fawaz, "Privacy against statistical inference," in *2012 50th annual Allerton conference on communication, control, and computing (Allerton)*. IEEE, 2012, pp. 1401–1408.

[29] W. Wang, L. Ying, and J. Zhang, "On the relation between identifiability, differential privacy, and mutual-information privacy," *IEEE Trans. Inf. Theory*, vol. 62, no. 9, pp. 5018–5029, 2016.

[30] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *Journal of artificial intelligence research*, vol. 4, pp. 129–145, 1996.

[31] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.

[32] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *25th USENIX Security Symposium*, 2016, pp. 601–618.

[33] V. Chandrasekaran, K. Chaudhuri, I. Giacomelli, S. Jha, and S. Yan, "Model extraction and active learning," *arXiv preprint arXiv:1811.02054*, 2018.

[34] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[35] D. Lopez-Paz, L. Bottou, B. Schölkopf, and V. Vapnik, "Unifying distillation and privileged information," *arXiv preprint arXiv:1511.03643*, 2015.

[36] J. H. Friedman, "Multivariate adaptive regression splines," *Ann. Stat.*, pp. 1–67, 1991.

[37] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[38] K. Hamidieh, "A data-driven statistical model for predicting the critical temperature of a superconductor," *Computational Materials Science*, vol. 154, pp. 346–354, 2018.

[39] K. T. Smith, D. C. Solmon, and S. L. Wagner, "Practical and mathematical aspects of the problem of reconstructing objects from radiographs," *Bulletin of the American Mathematical Society*, vol. 83, no. 6, pp. 1227–1270, 1977.

[40] F. Deutsch, "The angle between subspaces of a hilbert space," in *Approximation theory, wavelets and applications*. Springer, 1995, pp. 107–130.

VI. Appendix

We will use the following lemma. Let M, N be two closed subspaces in a Hilbert Space $\mathcal{H}$. Their Friedrichs angle is defined to be the number $0 \leq \theta_F \leq \frac{\pi}{2}$ such that

$$\cos\theta_F = \sup_{\substack{\mathbf{x}\in M\cap(M\cap N)^\perp, \mathbf{y}\in N\cap(M\cap N)^\perp \\ \|\mathbf{x}\|,\|\mathbf{y}\|\leq 1}} \mathbf{y}^{\mathrm{T}}\mathbf{x}. \tag{3}$$

Lemma 1. [39] Let $M_1, M_2, M_3, \ldots, M_k$ be closed subspaces in $\mathcal{H}$ with intersection $M = \bigcap_{i=1}^k M_i$. For $j = 1, 2, 3 \ldots, k$, we denote θ_F^j to be the *Friedrichs angle* between M_j and $\bigcap_{i=j+1}^k M_i$, then for any $x \in \mathcal{H}$ and any integer $n \geq 1$, we have

$$\|(P_k \ldots P_2 P_1)^n x - P_M x\| \leq c^n \|x - P_M x\| \tag{4}$$

where $c = \left(1 - \prod_{j=1}^{k-1} \sin^2\theta_F^j\right)^{1/2}$.

Proof of Theorem 1. We prove for the ordinary linear regression. The same technique can be extended to general additive models. For any design matrix $X \in \mathbb{R}^{n\times p}$, we define the projection matrix $P_X = X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}$ and its orthogonal $P_X^\perp = I_n - P_X$. We let $X = [X_A, X_B]$, with $X_A \in \mathbb{R}^{n\times p_1}$, $X_B \in \mathbb{R}^{n\times p_2}$ $(p_1 + p_2 = p)$, and $y \in \mathbb{R}^{n\times 1}$ be the corresponding labels. For simplicity, we use A, B to denote $span(X_A)$, $span(X_B)$ respectively. Also, we denote $\|\cdot\|$ to be the Euclidean norm and $\|\cdot\|_2$ to be the matrix operator norm.

Denote e_{orac} to be the residual obtained from the linear regression of y on X, i.e., $e_{orac} = y - \hat{y} = y - (X_A\hat{\beta}_a + X_B\hat{\beta}_b)$, where $[\hat{\beta}_a, \hat{\beta}_b]$ is the oracle least square estimator from all the data. Suppose that Alice holds data X_A and the label y, and Bob only has data X_B. Let e_i denote the residual at ith iteration and $e_0 = y$. Since they both use linear regression models, the residual e_k at kth iteration is:

$$e_k = (P_B^\perp P_A^\perp)^k e_0,$$

and we also have the following identity:

$$e_{orac} = P_{A\cup B}^\perp e_0 = P_{A^\perp\cap B^\perp} e_0.$$

By Lemma 1, for any integer $k \geq 1$, we have

$$\begin{aligned}
\|e_k - e_{orac}\| &= \left\| \left(P_B^\perp P_A^\perp\right)^k e_0 - P_{A^\perp\cap B^\perp} e_0 \right\| \leq c^k \|e_0 - P_{A^\perp\cap B^\perp} e_0\| \\
&= c^k \|e_0 - e_{orac}\| = (1 - \sin^2\theta_F)^{k/2} \|e_0 - e_{orac}\| \\
&= (\cos\theta_F)^k \|e_0 - e_{orac}\|
\end{aligned} \tag{5}$$

where $\cos\theta_F$ is the *Friedrichs angle* between $A^\perp$ and $B^\perp$. Since $\cos\theta_F = \cos\theta_F^\perp$ [40], and $\cos\theta_F < 1$ (since X has a full column rank), the error term will converge exponentially.

In the above arguments, we showed that e_k will converge to e_{orac} as k become large. Next we explicitly show the the aggregated coefficients obtained by Alice ad Bob will asymptotically approach the oracle least square estimators defined above. As a result, Alice will attain near-oracle performance from the assistance of Bob.

Proposition 1. Let $\hat{\beta}_a^k$, $\hat{\beta}_b^k$ be the coefficients obtained at the kth round of communication for Alice and Bob respectively, and $\hat{\beta}_a, \hat{\beta}_b$ be the oracle coefficients (defined as above). Then we have:

$$\begin{aligned}
\lim_{k\to\infty} \sum_{i=1}^k \hat{\beta}_a^i &= \hat{\beta}_a \\
\lim_{k\to\infty} \sum_{i=1}^k \hat{\beta}_b^i &= \hat{\beta}_b
\end{aligned} \tag{6}$$

Proof of Proposition 1. We proof for the case of Alice, i.e. $\lim_{k\to\infty} \sum_{i=1}^k \hat{\beta}_a^i = \hat{\beta}_a$. The similar technique can be used to prove Bob's case. From the procedure of assisted learning, the kth coefficient for Alice

is $(X_A^T X_A)^{-1} X_A^T (P_B^\perp P_A^\perp)^{k-1} y$, and we know $\hat{\beta}_a = (X_A^T P_B^\perp X_A)^{-1} X_A^T P_B^\perp y$ by some calculations. Then it suffices to show

$$(X_A^T P_B^\perp X_A)^{-1} X_A^T P_B^\perp = (X_A^T X_A)^{-1} X_A^T \left(\sum_{k=0}^{\infty} (P_B^\perp P_A^\perp)^k \right). \tag{7}$$

By *Gelfand's formula*, we have

$$\rho(P_B^\perp P_A^\perp) \leq \|P_B^\perp P_A^\perp\|_2, \tag{8}$$

where $\rho(\cdot)$ is the spectral radius (the largest absolute value of eigenvalues). From *Spectral Theorem*, we know that for any square matrix A, A is normal if and only if the operator norm equals the spectral radii. Therefore, we consider the following two cases.

Case 1: If $P_B^\perp P_A^\perp$ is normal, then

$$P_B^\perp P_A^\perp P_B^\perp = P_A^\perp P_B^\perp P_A^\perp \tag{9}$$

We just need to show that

$$X_A^T P_B^\perp = X_A^T P_B^\perp X_A (X_A^T X_A)^{-1} X_A^T \left(\sum_{k=0}^{\infty} (P_B^\perp P_A^\perp)^k \right) \tag{10}$$

Plugging (9) into the right hand side of (10), we have

$$\begin{aligned} X_A^T P_B^\perp X_A (X_A^T X_A)^{-1} X_A^T (\sum_{k=0}^{\infty} (P_B^\perp P_A^\perp)^k) &= X_A^T P_B^\perp P_A (I_n + P_B^\perp P_A^\perp + P_A^\perp P_B^\perp P_A^\perp + P_A^\perp P_B^\perp P_A^\perp + P_A^\perp P_B^\perp P_A^\perp + \cdots) \\ &= X_A^T P_B^\perp P_A + X_A^T P_B^\perp P_A P_B^\perp P_A^\perp = X_A^T P_B^\perp P_A + X_A^T P_B^\perp (I_n - P_A^\perp) P_B^\perp P_A^\perp \\ &= X_A^T P_B^\perp P_A + X_A^T P_B^\perp P_A^\perp + X_A^T P_B^\perp P_A^\perp P_B^\perp P_A^\perp. \end{aligned} \tag{11}$$

Since $X_A^T P_B^\perp P_A^\perp P_B^\perp P_A^\perp = X_A^T P_A^\perp P_B^\perp P_A^\perp = (P_A^\perp X_A)^T P_B^\perp P_A^\perp = 0$, then Eq. (11) reduces to

$$X_A^T P_B^\perp P_A + X_A^T P_B^\perp P_A^\perp = X_A^T P_B^\perp (P_A + P_A^\perp) = X_A^T P_B.$$

Therefore, Eq. (10) is correct and Case 1 holds.

Case 2: If $P_B^\perp P_A^\perp$ is not normal, then the equality in Eq. (8) will not hold. By a simple fact that $\|P_B^\perp P_A^\perp\|_2 \leq 1$, we have $\rho(P_B^\perp P_A^\perp) < 1$. By the property of *Neumann Series*, the following holds:

$$\sum_{t=0}^{\infty} (P_B^\perp P_A^\perp)^t = (I_n - P_B^\perp P_A^\perp)^{-1},$$

and $(I_n - P_B^\perp P_A^\perp)^{-1}$ exists.

We just need to show

$$(X_A^T X_A)^{-1} X_A^T (I_n - P_B^\perp P_A^\perp)^{-1} = (X_A^T P_B^\perp X_A)^{-1} X_A^T P_B^\perp, \tag{12}$$

by multiplying $(I_n - P_B^\perp P_A^\perp)$ on both sides of (12), it reduces to

$$(X_A^T X_A)^{-1} X_A^T = (X_A^T P_B^\perp X_A)^{-1} X_A^T P_B^\perp - (X_A^T P_B^\perp X_A)^{-1} X_A^T P_B^\perp P_A^\perp.$$

Since X_A is with full column rank, then $X_A X_A^T$ is invertible. Multiplying it on both sides, we have

$$(X_A^T X_A)^{-1} X_A^T X_A X_A^T = (X_A^T P_B^\perp X_A)^{-1} X_A^T P_B^\perp X_A X_A^T - (X_A^T P_B^\perp X_A)^{-1} X_A^T P_B^\perp P_A^\perp X_A X_A^T,$$

and it remains to show $X_A^T = X_A^T$, which is obviously true. Hence, Case 2 holds and we conclude the proof of Proposition 1. In conclusion, if Alice and Bob use linear regression models, then for a sufficiently large number of communications k, the oracle performance will be achieved and the error will decay exponentially.

In fact, the Theorem 1 above concerns a finite-sample result when the data size n remains fixed. The following result extends Theorem 1 to a probabilistic setting with random observations and varying n. Suppose that the data generating model is $y = \beta_a^\intercal x_a + \beta_b^\intercal x_a + \varepsilon$, where ε has zero mean and σ^2 variance, $x = [x_a, x_b] \in \mathbb{R}^p$ follows from a subGaussian distribution with zero mean and correlation matrix R, and x, ε are independent. Suppose that n independent observations $(y_i, x_{a,i})$ are available to Alice, and $(x_{b,i})$ are available to Bob, $i = 1, \ldots, n$. Let

$X = [x_1, \ldots, x_n]^{\mathrm{T}}$ denotes the design matrix centralizing all the data. Recall that $\mathcal{S}_a, \mathcal{S}_b$ denote the variable indices of Alice and Bob, respectively.

Corollary 1. Assume that the smallest eigenvalue of $X^{\mathrm{T}}X/n$ is almost surely lower bounded by a positive constant. Also assume that x is sub-Gaussian with a fixed covariance matrix, and $\mathbb{E}y^2 < \infty$. Then the final predictor of Alice $\tilde{y}_n^*$ satisfies $\mathbb{E}(\tilde{y}^* - y)^2 \to \sigma^2$ as $n \to \infty$, meaning that it is a consist predictor.

Proof of Corollary 1. Let $e_{n,k}$ and $e_{n,\text{orac}}$ denote the residual of Alice at step k of stage I, and the oracle residual by pulling all the data, respectively, where the subscript n highlights their dependence on the data size. Suppose there are k communications in Stage I. In stage II, suppose that the aggregated linear prediction function of Alice forms has a coefficient vector $\tilde{\beta}_{n,k}$; also suppose the oracle least square estimate by pulling the data is $\hat{\beta}_n$. It suffices to prove that $\tilde{\beta}_{n,k} - \hat{\beta}_n \to 0$ in probability as $n \to \infty$. By the subGaussian assumption, the *Friedrichs angle* between X_A and X_B, $\cos\theta_F$, is bounded away from 1 with probability at most $c_1 p^2 e^{-c_2 n t^2}$ for some constants c_1, c_2. Using Theorem 1 and the assumption on the smallest eigenvalue, there exists a constant c that

$$\|\tilde{\beta}_{n,k} - \hat{\beta}_n\|^2 \leq cn^{-1}\|X\tilde{\beta}_{n,k} - X\hat{\beta}_n\|^2 = cn^{-1}\|e_{n,k} - e_{n,\text{orac}}\|^2$$

which goes to zero in probability.

In the above corollary, it is possible that $p \to \infty$ and $k/p \to 0$ as $n \to \infty$, maintaining a high privacy for Bob since only a small fraction of column space is available to Alice.