PRIVACY-AWARE DISTRIBUTED GRAPH-BASED SEMI-SUPERVISED LEARNING

Başak Güler, A. Salman Avestimehr and Antonio Ortega

University of Southern California

ABSTRACT

This paper proposes a privacy-aware framework for distributed semi-supervised learning. In particular, we consider a semi-supervised learning problem where the training data is distributed among multiple data-owners, who wish to protect the privacy of their individual datasets from the other parties during training. We propose a novel framework for protecting the privacy of individual datasets while achieving good accuracy. Then, we characterize the privacy of our framework, by defining a metric that quantifies the number of candidate data points that are consistent with information shared by data-owners. The number of candidates (and thus the privacy) decreases as more information is shared between data-owners, leading to a privacy-utility (accuracy) trade-off. Our experiments show a significant increase in classification accuracy compared to local training, i.e., using the individual datasets only, while the complexity of our approach is significantly lower than that of other benchmarks, such as secure multi-party computing or homomorphic encryption.

Index Terms— Semi-supervised learning, privacy and security, distributed training

1. INTRODUCTION

Distributed learning provides an effective way to increase training performance, by allowing multiple data-owners to collaborate and learn from a large pool of data. On the other hand, the training data often carries sensitive information, such as healthcare records, financial transactions, or image and audio recordings. The privacy of such sensitive information should be protected from potential misuses, as failing to do so may raise serious ethical and legal concerns.

This paper considers a scenario in which multiple data-owners (clients) wish to collaborate to perform a semi-supervised task. Each party owns a local dataset, in which only a small number of data points are labeled and the remaining points are unlabeled. The goal for each party is to learn the unknown labels by using the information from the known labels and the unlabeled data points (feature vectors) and to do this using both data they own and data owned by others.

We focus on a graph-based semi-supervised learning (SSL) setup, in which training is carried out through label propagation on a similarity graph, where the edge weight between nodes representing two data points is a function of pairwise distances between those points [1–3]. Our goal is to allow training using data from multiple owners to be used, while minimizing the impact this has on privacy.

A straightforward approach to address this problem would be to perform the learning task at each client independently, using only the local datasets. This would achieve perfect privacy, as no information is shared between the clients, but may lead to lower classification accuracy if labeled data is scarce, as is often the case in SSL. Another possible approach would be to use *secure multi-party computation (MPC)* (e.g., [4–6]), where individual datasets would be secretly shared among data-owners, and then graph construction and label propagation algorithms would be performed on the secret shares. This approach has been investigated for the supervised learning setup [7–9], in which multiple data-owners jointly train a linear or logistic regression model, while keeping their individual datasets private from the other parties. At the end, all parties learn the final model while ensuring that their individual datasets are kept information-theoretically private. The MPC protocol, however, requires extensive communication and computation between the parties and the development of more efficient secure protocols remains an active area of research. A third alternative would be *homomorphic encryption (HE)* [10], which allows computations to be performed in the encrypted domain. This approach has also been studied for the supervised learning setup [11, 12]. Computing in the encrypted domain, however, requires extensive computational resources. In summary, existing frameworks can provide strong information-theoretic or computational privacy guarantees, while achieving good classification accuracy by training jointly across several datasets, but they require significant computation or communication overhead, which hinders their applicability in large-scale systems. Another line of work [13] considers a distributed SSL setup in which the pairwise distances are computed in a privacy-preserving manner, i.e., without revealing the individual data points. However, as we show in our paper, the distances themselves can reveal extensive information about the data points through a simple interpolation strategy and breach privacy. Hence, the information revealed from the distances should be limited to protect the privacy of the datasets.

In this paper, we propose PSSL, a lightweight privacy-preserving framework for distributed SSL. PSSL allows information to be shared openly between clients, but the amount and choice of shared information are limited in order to preserve privacy. Since this information is shared openly there is no encryption or secure communication overhead, which significantly reduces complexity with respect to existing methods. Following the graph SSL perspective, each client builds a local graph with its own data, and then reveals the similarity between a few of its data points and those of others (essentially creating a few connections across private datasets).

The key intuition in our work is that revealing the similarity between points (a function of distance in a high dimensional space) is not that informative if the number of connections created is small. Revealing the similarities between more and more data points increases the information leakage, but also improves the training performance. Our framework is based on a trade-off between i) the information that is leaked by the connections of each data point and ii) the classification accuracy. We create a few informative inter-client connections within a privacy-preserving protocol to improve the classification accuracy significantly while protecting the privacy of the individual data points. We then analyze the privacy-utility trade-off of our system, by introducing a novel measure to quantify how much information is leaked from the connections for each data point, in terms of the number of neighbors as well as the similarity between connected data points. In particular, we characterize the privacy of our framework by defining a metric that quantifies the number of candidate data points that are consistent with information shared by data-owners. The number of candidates (and thus the privacy) decreases as more information is shared between data-owners,

978-1-7281-0824-7/19/$31.00 ©2019 IEEE

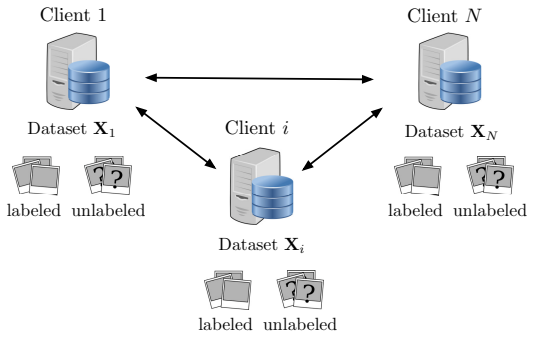

Fig. 1. Illustration of the distributed SSL setup with N data-owners (clients). Client $i \in [N]$ owns a dataset $\mathbf{X}_i$ (in this example, a collection of images), in which a small number of data points are labeled and the remaining ones are unlabeled. The clients collaborate to run an SSL task to estimate their unknown labels.

leading to a privacy-utility (accuracy) trade-off. We also compare the computational complexity of PSSL with cryptographic benchmarks, such as secure MPC and HE, and show that PSSL can reduce the complexity substantially.

Through numerical experiments, we show that PSSL can improve classification accuracy significantly compared to local SSL (i.e., using the individual dataset only). Moreover, we visually demonstrate that information leakage from each user is minimal.

Note that one could also approach this problem using differential privacy, which adds noise to protect the privacy of personally identifiable information, without changing the training results significantly [14–19]. Although our focus is on understanding how much privacy can be provided specifically by controlling the structure of the similarity graph, it may be possible in principle to combine our techniques with differential privacy approaches.

Notation. In the remainder of the paper, x is a scalar variable, $\mathbf{x}$ is a vector, and $\mathcal{X}$ is a set with cardinality $|\mathcal{X}|$. $\mathbf{A}$ is a matrix with $\mathbf{a}_i$ denoting its i^{th} row. $[N]$ denotes the set $\{1, \ldots, N\}$.

2. THE PROPOSED PRIVACY-AWARE SEMI-SUPERVISED LEARNING FRAMEWORK

We consider a distributed SSL scenario with N data-owners (clients) as depicted in Figure 1. Client $i \in [N]$ owns an individual dataset consisting of M data points with F features, represented by an $M \times F$ matrix $\mathbf{X}_i = [\mathbf{x}_{i1}^\top \ldots \mathbf{x}_{iM}^\top]^\top$. Each data point belongs to one of L classes, but only a small subset of the data points are labeled. The labels are represented by an $M \times L$ matrix $\mathbf{Y}_i = [\mathbf{y}_{i1}^\top \ldots \mathbf{y}_{iM}^\top]^\top$, where $\mathbf{y}_{ij} = \mathbf{e}_l$ if data point $\mathbf{x}_{ij}$ is labeled with class $l \in [L]$, and $\mathbf{y}_{ij} = \mathbf{0}$ if it is not labeled. Vector $\mathbf{e}_l$ is the standard unit vector whose l^{th} element is 1 and the remaining elements are 0.

The clients wish to learn the missing labels in their individual datasets. In order to improve their accuracy, clients can collaborate and use each other's labeled and unlabeled data while protecting the privacy of their individual dataset [1]. We also assume that clients do not collude, i.e., they do not share information beyond what the protocol requires them to. Our distributed SSL framework consists of two main phases: *1) private graph construction*, where clients construct a similarity graph based on distances between data points, while preserving the privacy of their individual data, and *2) label propagation*, where clients jointly process information in the distributed similarity graph to learn the missing labels in their individual datasets. These two phases are now described in detail.

[1]Our focus is on protecting the privacy of the actual data points (e.g., images), and not the labels (e.g., whether an image represents a dog or a cat).

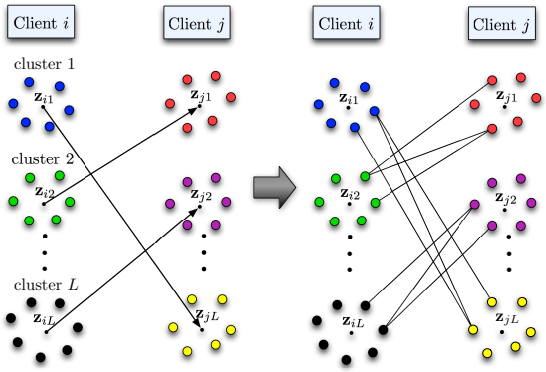

Fig. 2. Construction of the inter-client connections between clients i and j. Each client initially partitions its own dataset into L clusters. Then, within a secure MPC, for each cluster from client i, the closest cluster from client j is identified, using the cluster centers (clients do not learn the location of the cluster centers). Then, R random connections are created between each cluster pair.

Private Graph Construction. In this phase, clients construct a similarity graph in which similar data points are connected by an edge. We define the distance between two data points as,

$$d_{ij}(u, v) := \|\mathbf{x}_{iu} - \mathbf{x}_{jv}\| \text{ for } i, j \in [N], \quad u, v \in [M], \tag{1}$$

where data points $\mathbf{x}_{iu}$ and $\mathbf{x}_{jv}$ belong to clients i and j, respectively. The weight of an edge, defined as $w_{ij}(u, v) := e^{-d_{ij}(u,v)^2/\varepsilon}$ for a given $\varepsilon > 0$, quantifies the similarity between two data points. Two types of edges are created, the first correspond to "local" connections, i.e., between two nodes corresponding to data points available to a given client, whereas the second are inter-client connections linking data points stored by two different clients. The connections that affect privacy are the inter-client ones. Thus, a good graph should have only a few informative inter-client connections, such that those nodes that are connected should not be too similar to one another. This way one can achieve improved classification performance while preserving privacy. While this principle is general, we next describe a specific approach for graph construction.

(Local connections) Initially, client $i \in [N]$ locally constructs a mutual K-nearest neighbor (K-NN) graph using its individual dataset $\mathbf{X}_i$. In this graph, each data point $\{\mathbf{x}_{iu}\}_{u\in[M]}$ corresponds to a distinct node. An edge is drawn between two distinct nodes u and v if and only if node u is among the K closest data points to node v and vice versa, with respect to the distance measure $d_{ii}(u, v)$ from (1).

(Inter-client connections) Clients create connections between their data points and the data points that belong to other clients. As illustrated by Figure 2, client $i \in [N]$ first locally partitions its data points $\{\mathbf{x}_{iu}\}_{u\in[M]}$ into L clusters through k-means clustering with J iterations [20] (by setting $k = L$). The corresponding cluster centers are denoted by $\{\mathbf{z}_{i1}, \ldots, \mathbf{z}_{iL}\}$. Then, each pair of clients $i, j \in [N]$ carry out the following computations within a secure two-party MPC protocol, through pairwise computations in a decentralized manner (Note that this is a much lighter usage of MPC compared to multi-party MPC, as it does not require full cooperation between N clients, and is not used during training.). First, for each cluster $l \in [L]$ from client i, the closest cluster from client j is found by computing a distance between cluster centers,

$$t = \arg\min_{t' \in [L]} \|\mathbf{z}_{il} - \mathbf{z}_{jt'}\|. \tag{2}$$

Second, R random connections are created between the two clusters (l and t) with the condition that, node u from client i can be connected to node v from client j as long as,

Algorithm 1 Private Graph Construction

Input: $\{\mathbf{X}_i\}_{i\in[N]}$, L, and parameters K, R, d_{min}, d_{max}, λ
Output: Similarity graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Local connections:
for client $i \in [N]$ **do**
  Construct a mutual K-NN graph locally using $\mathbf{X}_i$.
  Let $\mathcal{V}_i$ denote the corresponding set of nodes.
Inter-client connections:
for client $i \in [N]$ **do**
  Partition $\{\mathbf{x}_{iu}\}_{u\in[M]}$ into L clusters via k-means clustering.
for client $i \in [N]$ **do**
  for client $j \in [N]\backslash\{i\}$ **do**
    Within a secure two-party MPC protocol:
    for cluster $l \in [L]$ **do**
      Find the closest cluster t from client j using (2).
      Create R random connections satisfying (3) and (4) between clusters l and t.
    for each node v of client j selected in Step 10 **do**
      if equation (7) has a single solution **do**
        Discard all connections from v to $\mathcal{N}_{ji}(v)$.
      else
        Set $\mathcal{V}_j \leftarrow \mathcal{V}_j\backslash\{v\}$ and $\mathcal{V}_i \leftarrow \mathcal{V}_i\backslash\mathcal{N}_{ji}(v)$.
    Repeat Steps 11-15 with j and i interchanged.

$$i)\ \ d_{min} \leq d_{ij}(u,v) \leq d_{max} \tag{3}$$

$$ii)\ \ \max\left(|\mathcal{N}_{ij}(u)|, |\mathcal{N}_{ji}(v)|\right) \leq \lambda \tag{4}$$

for a given set of parameters $d_{min}, d_{max}, \lambda$, where $\mathcal{N}_{ij}(u)$ represents the set of neighbors from node u of client i to client j (similarly, $\mathcal{N}_{ji}(v)$ is the set of neighbors from node v of client j to client i). Third, any client, i, checks whether the information to share with another client, j, would allow j to identify exactly the position of a data point owned by i and connected to points from j. This is checked based on the pairwise distances using an approach that will be described in Section 3. If the check fails, the inter-client connections corresponding to that node are discarded. Finally, the corresponding weights $w_{ij}(u,v)$ are revealed to clients i and j. Algorithm 1 demonstrates the graph construction procedure.

The two conditions in (3) and (4) correspond to the trade-off between utility and privacy provided by our system. As will be discussed in Section 3, condition (3) ensures that the revealed inter-client neighbors are not too close or too far. A balance needs to be achieved between ensuring that privacy is preserved (d_{min} large enough) while creating connections that are useful (d_{max} should not be too large). Condition (4) ensures that a node is not connected to too many neighbors. This is due to the fact that locating any data point can be viewed as an interpolation problem using the knowledge of the location of its neighbors and the corresponding distances. As will be shown in Section 3 inter-client connections allow a client to determine the location of data from another client within a multi-dimensional sphere, whose dimension decreases as the number of neighbors increases and the distance to the neighbors decreases.

As the computations in this phase are carried out within a secure MPC protocol, clients only learn the final weights $w_{ij}(u,v)$, and beyond that they learn no information about the actual data points belonging to the other clients, as well as the cluster centers. In other words, client i only learns that node u is connected to some node v with weight $w_{ij}(u,v)$ but no more information about the actual data point $\mathbf{x}_{jv}$. Similarly, client j only learns that node v is connected to some node u with weight $w_{ij}(u,v)$ but nothing more about $\mathbf{x}_{iu}$. The privacy protection of this secure weight computation step follows from the privacy guarantees of the secure MPC protocol, and can be based on information-theoretic or computational assumptions, depending on the specific implementation [4–6].

Distributed Label Propagation. Learning is carried out through a distributed label propagation algorithm, built on the centralized label-propagation protocol from [2]. This is an iterative process where, at each iteration, clients update the current estimation of their labels by utilizing the information received from their neighbors.

Initially, client $i \in [N]$ constructs an $M \times M$ similarity matrix $\mathbf{W}_{ii}$ with the $(u,v)^{th}$ element being equal to $w_{ii}(u,v)$ if node u is locally connected to node v, and 0 otherwise. In addition, for each $j \in [N]\backslash\{i\}$, client i constructs an $M \times M$ matrix $\mathbf{W}_{ji}$ with the $(u,v)^{th}$ element being equal to $w_{ji}(u,v)$ if node u from client j is connected to node v of client i, and 0 otherwise. Note that these weights were calculated in the graph construction phase. Finally, client i defines an $M \times L$ matrix $\mathbf{A}_i^{(t)}$ to hold the label estimations at iteration t, and initializes them as $\mathbf{A}_i^{(0)} = \mathbf{Y}_i$. Each row represents a probability distribution on L classes, that is, row $u \in [M]$ specifies the probability that node u belongs to each of the L classes.

At iteration t, client i computes $\mathbf{W}_{ji}\mathbf{A}_i^{(t)}$ and sends it to client j, for each $j \in [N]$. Accordingly, client i receives the computation results $\mathbf{W}_{ij}\mathbf{A}_j^{(t)}$ from clients $j \in [N]$. Client i then locally updates its label estimations as follows,

$$\mathbf{A}_i^{(t+1)} = \alpha\mathbf{D}_i^{-1}\sum_{j\in[N]}\mathbf{W}_{ij}\mathbf{A}_j^{(t)} + (1-\alpha)\mathbf{Y}_i \tag{5}$$

where parameter $\alpha \in (0,1)$ quantifies the amount of information carried from the neighbors versus the initial labels. $\mathbf{D}_i$ is a diagonal matrix formed by client i that holds the weighted degree of node u,

$$deg(u) = \sum_{j\in[N]}\sum_{v\in\mathcal{N}_{ij}(u)} w_{ij}(u,v) \tag{6}$$

at diagonal u. At the end of T iterations, client i computes the labels of its data points using $\mathbf{A}_i^{(T)}$. Specifically, node $u \in [M]$ is assigned to the label with the highest probability, using the estimated probabilities in row u of $\mathbf{A}_i^{(T)}$. The convergence of the algorithm follows from similar arguments in [2], by letting $\mathbf{Y} = [\mathbf{Y}_1^T \dots \mathbf{Y}_N^T]^T$ and $\mathbf{S} \triangleq \mathbf{D}^{-1}\mathbf{W}$ such that, $\mathbf{D} = diag(\mathbf{D}_1, \dots, \mathbf{D}_N)$ and the $(i,j)^{th}$ element of $\mathbf{W}$ is equal to $\mathbf{W}_{ij}$ for $i,j \in [N]$. As can be observed from (5), we use the random walk Laplacian $\mathbf{D}^{-1}\mathbf{W}$ as our regularizer, unlike [2] where the regularizer is the normalized Laplacian $\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$. Our choice of using the random walk Laplacian is due to the distributed nature of our algorithm, that normalization using the random walk Laplacian can be performed locally by each client as it only requires the knowledge of $\mathbf{D}_i$ at client i, since the normalization of node u at client i is given by $w_{ij}(u,v)/deg(u)$. On the other hand, the normalized Laplacian requires $\mathbf{D}_i$ as well as $\mathbf{D}_j$ for $j \in [N]\backslash\{i\}$, which belong to other clients, since in this case the normalization of node u at client i is given by $w_{ij}(u,v)/\sqrt{deg(u)deg(v)}$. This may in turn reveal additional information about them.

Lastly, for additional privacy protection against potential leakage of information through intermediate computations during the label propagation phase, one can incorporate conventional secure computation techniques, such as secure multi-party MPC [4, 5] or HE [10], to perform the computations in (5).

3. PRIVACY ANALYSIS

Our privacy analysis is based on an inference problem, where the adversary tries to guess a specific data point given the neighboring data points as well as their relative distances. In the following, to

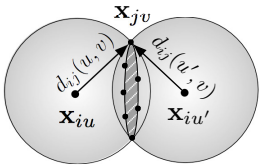

Fig. 3. Locating an unknown data point $\mathbf{x}_{ju}$ in $\mathbb{R}^3$, using the neighboring points $\mathbf{x}_{iu}$ and $\mathbf{x}_{iu'}$ with corresponding distances $d_{ij}(u,v)$ and $d_{ij}(u',v)$, respectively. The candidate locations for $\mathbf{x}_{ju}$ are the points on the surface of a 2-dimensional sphere (the shaded circle), which is the intersection of two 3-dimensional spheres, centered at $\mathbf{x}_{iu}$ and $\mathbf{x}_{iu'}$ with radii $d_{ij}(u,v)$ and $d_{ij}(u',v)$, respectively.

refer to a specific element from the dataset $\mathbf{X}_i$, we use the terms data point and node interchangeably. Suppose that client i attempts to guess the value of node $\mathbf{x}_{jv}$ for some $v \in [M]$ belonging to client j. Then, the problem of locating node $\mathbf{x}_{jv}$ corresponds to solving a system of quadratic equations:

$$\|\mathbf{x}_{iu} - \mathbf{x}_{jv}\|^2 = d_{ij}(u,v)^2 \quad \text{for all } u \in \mathcal{N}_{ji}(v). \tag{7}$$

where $|\mathcal{N}_{ji}(v)| \leq \lambda$. Refer to Figure 3 for an illustration. By definition, the system (7) has at least one feasible solution, which is the actual point $\mathbf{x}_{jv}$. Our privacy measure is equal to the number of candidate data points for any $\mathbf{x}_{jv}$, which is the number of solutions to (7) and is parameterized by the quantity d_{min}. That is, increasing d_{min} increases the pairwise distance $d_{ij}(u,v)$ and the number of solutions to (7). As a result, the adversary has to make a guess from a larger set of candidate solutions, which is the basis of our privacy-preserving algorithm. We now present how the number of solutions increases with respect to d_{min}, by characterizing the solution space of (7) for two feature domains: Euclidean and integer domains. Initially, we consider the Euclidean domain, i.e., each data point resides in the Euclidean space of dimension F.

Proposition 1. *If $\mathbf{X}_i \in \mathbb{R}^{M\times F}$ for $i \in [N]$, then, the solution to* (7) *is either: i) A point on the surface of a multi-dimensional sphere with dimension no less than $F - \lambda$, ii) A single point.*

Proof. The proof follows from the observation that solving (7) is equivalent to finding the intersection of λ multi-dimensional spheres where the dimension of each sphere is F. Then, the proposition is a direct result of the intersection of spheres problem [21]. □

Whether the solution is a single point can be checked by a simple QR-decomposition procedure [21]. In Algorithm 1, this verification is implemented within the MPC protocol between pairs of clients, while forming the inter-client edges. Proposition 1 characterizes the solution space if a client tries to guess the value of a data point from another client, when the data points reside in a Euclidean space.

Next, we consider the problem of characterizing the number of candidate solutions when the domain is discrete, as in real-world scenarios such as imaging, where pixel values are integer. Thus, we now consider the case when the data points are realized from a finite domain of non-negative integers. For this case, characterizing the solution space, in particular, the number of solutions to (7), is an open problem in general. In the following, we characterize the number of solutions for this setup by focusing on the special case $\lambda = 1$. This corresponds to the scenario in which any node is linked to at most one other node from another client. In other words, the only information that client i has about some node $\mathbf{x}_{jv}$ that belongs to client j is that it resides on the surface of an F-dimensional sphere with radius $d_{ij}(u,v)$ centered at $\mathbf{x}_{iu}$ (which is client i's own data point). In the sequel, we identify the relation between the number of candidate points on this sphere with respect to the relative distance (radius) $d_{ij}(u,v)$. Specially, we demonstrate how the number of candidate points grow as the pairwise distances increase. In Section 4, we demonstrate that this special case can achieve significant classification accuracy. We first state a useful lemma.

Lemma 1. *(Sum of Integer Squares [22]) Consider the equation,*

$$z_1^2 + \ldots + z_n^2 = d, \tag{8}$$

for some $\mathbf{z} = (z_1, \ldots, z_n)$ and n. Let $r_n(d)$ denote the number of different solutions to (8)*. If the domain of* (8) *is the set of all integers,*

$$r_n(d) = p_n(d) + O(d^{n/4}). \tag{9}$$

$p_n(d)$ is called the singular series $p_n(d) = \frac{\pi^{n/2}}{\Gamma(n/2)} d^{n/2-1}\rho(n,d)$, in which the first term $\frac{\pi^{n/2}}{\Gamma(n/2)}$ is independent of d, and the last term can be bounded by two constants $c_1 \leq \rho(n,d) \leq c_2$ where c_1 and c_2 are independent of d. Hence, for large d,

$$r_n(d) \sim d^{n/2}. \tag{10}$$

Proof. This lemma follows from standard results in number theory, and is available in [22]. □

Corollary 1. *The number of solutions from* (10) *in Lemma 1 includes both positive and negative integer solutions. One can remove the effect of potential sign differences as,*

$$r_n(d) \sim d^{n/2}/2^n, \tag{11}$$

to approximate the minimum number of solutions corresponding to distinct $\|\mathbf{z}\|^2 = (z_1^2, \ldots, z_n^2)$ sequences.

Next, consider the system in (7) with $\lambda = 1$ and $\mathbf{X}_i \in \mathcal{X}^{M\times F}$ for $i \in [N]$, where $\mathcal{X} = \{0, 1, \ldots, s\}$ is a finite set of $|\mathcal{X}| = s + 1$ non-negative integers. For this case, (7) reduces to a single equation, hence one can treat it as a sum of squares problem by setting $n = F$, $d = d_{ij}(u,v)^2$, and $\|\mathbf{z}\|^2 = \|\mathbf{x}_{iu} - \mathbf{x}_{jv}\|^2$ in (8). Unlike Lemma 1, however, the domain is now finite, and client i can use this information along with the data point $\mathbf{x}_{iu}$ to remove the infeasible solutions. The following lemma characterizes the structure of feasible and unfeasible solutions, which will be useful in our analysis.

Lemma 2. *Consider two vectors $\mathbf{a} \in \mathcal{X}^n$ and $\mathbf{b} \in \mathcal{X}^n$ where $\mathcal{X} = \{0, 1, \ldots, s\}$, and define*

$$\|\mathbf{a} - \mathbf{b}\|^2 = (a_1 - b_1)^2 + \ldots + (a_n - b_n)^2 = d. \tag{12}$$

Let $\mathbf{z} = (z_1, \ldots, z_n)$ be any solution to (8)*. If,*

$$z_i^2 \leq \lceil s/2 \rceil^2 \quad \forall i \in [N], \tag{13}$$

then, for any given $\mathbf{a} \in \mathcal{X}^n$, there exists some $\mathbf{b} \in \mathcal{X}^n$ that satisfy (12)*. In this case, we say that $\mathbf{z}$ is a feasible solution for any $\mathbf{a} \in \mathcal{X}^n$. Otherwise, if*

$$z_i^2 > \lceil s/2 \rceil^2 \quad \text{for some } i, \tag{14}$$

then, there exists $\mathbf{a} \in \mathcal{X}^n$ for which no $\mathbf{b} \in \mathcal{X}^n$ satisfy (12)*. In this case, we say that $\mathbf{z}$ is unfeasible for some $\mathbf{a} \in \mathcal{X}^n$.*

Proof. Suppose (13) holds. Then,

$$b_i = \begin{cases} |z_i| + a_i & \text{if } a_i < |z_i| \\ a_i - |z_i| & \text{if } a_i \geq |z_i| \end{cases} \tag{15}$$

is a feasible solution to (12), since $b_i \in \mathcal{X}$ for all $i \in [N]$. Next, suppose that (14) holds for some $i \in [N]$. Let $a_i = \lceil s/2 \rceil$ and observe that $0 \leq (a_i - b_i)^2 \leq \lceil s/2 \rceil^2$, since $0 \leq b_i \leq |\mathcal{X}|$. Hence, (14) cannot be satisfied for any $b_i \in \mathcal{X}$. Accordingly, $\mathbf{z}$ is not a feasible solution to (12). □

As a result, the actual number of feasible solutions may be much smaller than (10) when the domain is finite. We handle this problem by converting a given solution of (8) in a lower dimensional space (with dimension $n < F$) into a feasible solution in the higher dimensional space (with dimension F). This enables us to provide a lower bound on the number of solutions when the dimension of the feature space is sufficiently large. We next state our main theorem.

Theorem 1. *Let $\mathbf{X}_i \in \mathcal{X}^{M\times F}$ for $i \in [N]$, where the set $\mathcal{X} = \{0, 1, \ldots, s\}$ is a finite integer set, and $\lambda = 1$. Then, for guessing any data point, there are $\sim (d_{min}/2)^n$ solutions to* (7)*, where*

$$n = \left\lfloor F \Big/ \left(\frac{d_{max}^2}{\lceil s/2 \rceil^2} + 4 \right) \right\rfloor. \tag{16}$$

Remark 1. Theorem 1 shows that the number of candidate solutions to the inference problem increases as the minimum distance between the data points (d_{min}) or the dimension of the feature space (F) increase (for a fixed $\mathcal{X}$ and d_{max}). This constitutes the basis of our privacy criteria, in particular, one can improve privacy protection by reducing the number of neighbors at each node, by increasing the number of feature dimensions, or by increasing the pairwise distances, i.e., so that only increasingly distant neighbors are revealed.

Proof. Let $\mathbf{z}$ be any solution to (8) when $d = d_{ij}(u, v)^2$ and $n < F$. Then, $\mathbf{z}$ satisfies one of the following two cases:

Case 1: $z_i^2 \leq \lceil s/2 \rceil^2$ for all $i \in [n]$. In this case, one can convert $\mathbf{z}$ to a solution in (7). First, note that (7) represents a sum of F terms, where each term is an integer square. Next, select n random indices from (7) and set each term equal to a distinct element z_i^2 for $i \in [n]$. Set the remaining $F - n$ terms to 0. Observe from Lemma 2 that this is a feasible solution, since for any $\mathbf{x}_{iu}$, there exists a data point $\mathbf{x}_{iv}$ within the domain $\mathcal{X}$ that satisfies all the equality conditions.

Case 2: $z_i^2 > \lceil s/2 \rceil^2$ for some $i \in [n]$. In this case, we can convert $\mathbf{z}$ to a solution in (7) as follows. First, note that, $z_i^2 \leq \left\lfloor \frac{d_{ij}(u,v)^2}{\lceil s/2 \rceil^2} \right\rfloor \lceil s/2 \rceil^2 + c$ where c is a constant such that $c < \lceil s/2 \rceil^2$. It then follows from Lagrange's four square theorem [23] that c can be written as a sum of at most 4 integer squares. Therefore, each z_i^2 can be written as a sum of at most $\left\lfloor \frac{d_{ij}(u,v)^2}{\lceil s/2 \rceil^2} \right\rfloor + 4$ squares whose value is no greater than $\lceil s/2 \rceil^2$. Since there are at most n such z_i^2 terms, any solution $\mathbf{z}$ can be written as a sum of at most $n\left(\left\lfloor \frac{d_{ij}(u,v)^2}{\lceil s/2 \rceil^2} \right\rfloor + 4 \right)$ squares whose value is no greater than $\lceil s/2 \rceil^2$, after which we can use Lemma 2. Therefore, whenever

$$F \geq n \left(\left\lfloor \frac{d_{ij}(u, v)^2}{\lceil s/2 \rceil^2} \right\rfloor + 4 \right), \tag{17}$$

then any solution $\mathbf{z}$ of (8) can be converted to a solution of (7), by setting $n\left(\left\lfloor \frac{d_{ij}(u,v)^2}{\lceil s/2 \rceil^2} \right\rfloor + 4 \right)$ random indices equal to the corresponding terms and the remaining $F - n$ terms equal to 0. Hence, for any n satisfying (17), from Corollary 1, there are $(d_{ij}(u, v)/2)^n$ distinct solutions to (7). Finally, note that $d_{min} \leq d_{ij}(u, v) \leq d_{max}$ from (3). Therefore, if $F \geq n\left(\frac{d_{max}^2}{\lceil s/2 \rceil^2} + 4 \right)$, then, there are at least $\sim (d_{min}/2)^n$ candidate solutions for guessing any single data point. Then, (16) follows from selecting the largest feasible n. □

4. EXPERIMENTS

We now investigate the performance of the PSSL framework. To do so, we consider a distributed SSL task for image classification on the MNIST dataset [24], carried out by $N = 10$ clients. Each training sample in the dataset is an 28×28 image, converted into a vector of size $F = 784$. Pixel values are from $\mathcal{X} = \{0, \ldots, 255\}$. From the dataset, we extract the images for digits $1, 2, 3, 4$ (setting $L = 4$), and assign $M = 500$ images to each client randomly, with no duplicate assignments. Within each dataset, $r = 0.01$ fraction of samples (randomly selected) are labeled. The parameters for the edge weights are selected as $\varepsilon := 1/\gamma$ where $\gamma = 1.25$, and for label propagation we set $\alpha = 0.99$, in accordance with [2]. The training is done over $T = 20$ iterations. We then compare the PSSL framework from Section 2 with the following two baselines.

Table 1. Average classification accuracy

Protocol	Accuracy (%)	Bound of Theorem 1
CSSL	95.22	
PSSL ($d_{min} = 3s$)	80.34	1.46×10^5
PSSL ($d_{min} = 5s$)	79.96	4.06×10^5
PSSL ($d_{min} = 7s$)	74.98	7.96×10^5
LSSL	64.62	

1. Centralized SSL (CSSL): This is the conventional centralized SSL scenario in which the collection of individual datasets is treated as a single dataset, and a single mutual K-NN graph is constructed to identify the similarities between data points. Then, the label propagation algorithm from [2] is implemented on the resulting graph to learn the unknown labels. This protocol defines our baseline for the best performance in the graph-based learning scenario.

2. Local SSL (LSSL): In this scenario, each client performs the learning task by using its local dataset only, without collaborating with other clients. Specifically, each client first constructs a mutual K-NN graph locally using its individual dataset and then carries out the label propagation protocol from [2] using its local graph. In doing so, clients do not communicate with each other, and therefore no information is shared between them. Since all computations are performed locally, this scenario achieves perfect privacy.

Table 1 presents the average classification accuracy (over N clients) of our PSSL framework compared with the two benchmarks. We also present the lower bound $(d_{min}/2)^n$ from Theorem 1 with n chosen according to (16), which measures the privacy protection of the system (related to the number of candidates for guessing each data point). In all three scenarios, we let $K = 10$ for the mutual K-NN graph construction. For PSSL, we set the system parameters as $M = 1$, $d_{max} = 9s$, and $R = 80$, and present the change in the classification accuracy as the minimum distance d_{min} allowed between the neighboring nodes is increased. This represents the trade-off between the utility and privacy provided by our system. Specifically, as discussed in Section 3, in this case, the information available at client i about a data point $\mathbf{x}_{jv}$ belonging to client j consists of the location of at most a single neighboring data point $\mathbf{x}_{iu}$ and the distance $d_{ij}(u, v) \geq d_{min}$ between the two. As the minimum distance between the neighbors is increased, the number of potential candidates for guessing each node increases, which then increases the privacy protection of the system. On the other hand, the increased distances between neighboring nodes reduce the classification accuracy, hence the utility provided by the learning mechanism. In Figure 4, we illustrate the images corresponding to the neighboring data points as the minimum distance allowed between them is increased gradually. In particular, we plot the first 100 images known by client 1 and the corresponding neighbor from client 2. We visually observe that as the minimum distance d_{min} increases, the neighboring images become less and less similar to the actual image.

One can construct a global K-NN graph within a cryptographic computation protocol such as multi-party MPC or HE. Such a protocol, however, would require $O((NM)^2)$ secure computations. This limits their scalability to large systems, as secure computations require either extensive communication and coordination between the parties (the communication overhead of the well-known multi-party MPC from [5] is $O(N^2)$, i.e., quadratic in the number of clients), or computation to be performed in the encrypted domain (in HE, the size of the encrypted data is much larger than the original data, as its privacy protection depends on the size of the encrypted data), which

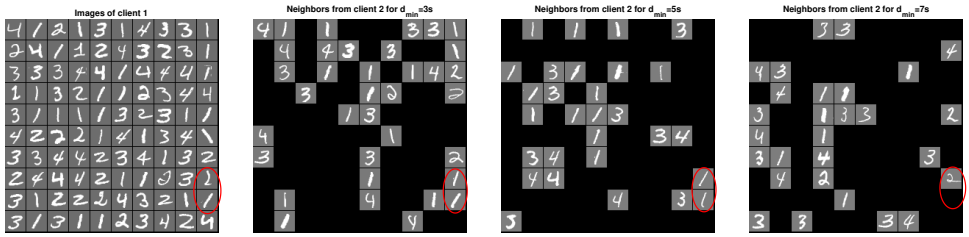

Fig. 4. Illustration of the first 100 images from client 1 and the corresponding neighbor from client 2, as the minimum distance d_{min} is increased. A black square denotes that the image from client 1 has no neighbors at client 2. The highlighted images demonstrate that when the distance is small, neighboring images look similar, whereas they start to look less similar as the distance becomes larger.

Table 2. Comparison of storage (per client), communication, and computation overhead of: 1) Centralized SSL with (multi-party) MPC, 2) Centralized SSL with HE, 3) Local SSL, and 4) PSSL. In HE, storage, communication, and computation is done over encrypted data, whose size is much larger than the original data size.

Protocol	Storage	Communication	Computation
CSSL-MPC	$O(MN)$	$O((M^2N^4)$	$O((MN)^2)$
CSSL-HE	$O(MN)$	$O(MN^2)$	$O((MN)^2)$
LSSL	$O(M)$	0	$O(M^2)$
PSSL	$O(M)$	$O(NLR)$	$O(\max\{M^2,MLJ,NLR\})$

can lead to many orders of magnitude slowdown. A comparison of the overhead incurred by these approaches is given in Table 2.

5. CONCLUSION

We have considered a distributed learning scenario in which multiple data-owners wish to jointly run a semi-supervised learning task without revealing their individual datasets to each other. We propose a lightweight privacy-aware learning framework and analyze its privacy protection. Numerical results demonstrate that our framework can achieve good learning accuracy while being privacy-preserving.

6. REFERENCES

[1] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using Gaussian fields and harmonic functions," in *Int. Conf. on Machine Learning*, 2003, pp. 912–919.

[2] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," *Adv. in Neural Inf. Proc. Sys.*, pp. 321–328, 2004.

[3] A. Gadde, A. Anis, and A. Ortega, "Active semi-supervised learning using sampling theory for graph signals," in *ACM KDD*, 2014, pp. 492–501.

[4] A. C. Yao, "Protocols for secure computations," in *IEEE Symp. on Foundations of Comp. Science*, 1982, pp. 160–164.

[5] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," *ACM Symp. on Theory of Comp.*, pp. 1–10, 1988.

[6] Z. Beerliová-Trubíniová and M. Hirt, "Perfectly-secure MPC with linear communication complexity," in *Theory of Cryptography Conference*. Springer, 2008, pp. 213–230.

[7] Payman Mohassel and Yupeng Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *IEEE Symp. on Security and Privacy*, 2017, pp. 19–38.

[8] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," *IEEE Symp. on Sec. & Priv.*, 2013.

[9] J. So, B. Guler, A. S. Avestimehr, and P. Mohassel, "CodedPrivateML: A fast and privacy-preserving framework for distributed machine learning," *arXiv:1902.00641*, 2019.

[10] C. Gentry and D. Boneh, *A fully homomorphic encryption scheme*, vol. 20, Stanford University, Stanford, 2009.

[11] T. Graepel, K. Lauter, and M. Naehrig, "ML confidential: Machine learning on encrypted data," in *Int. Conf. on Inf. Security and Cryptology*, 2012, pp. 1–21.

[12] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," *Int. Conf. on Machine Learning*, 2016.

[13] R. Fierimonte, S. Scardapane, A. Uncini, and M. Panella, "Fully decentralized semi-supervised learning via privacy-preserving matrix completion," *IEEE Trans. on Neural Net. and Learning Sys.*, vol. 28, no. 11, pp. 2699–2711, 2016.

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

[15] K. Chaudhuri and C. Monteleoni, "Privacy-preserving logistic regression," *Adv. in Neural Inf. Proc. Sys.*, pp. 289–296, 2009.

[16] A. Rajkumar and S. Agarwal, "A differentially private stochastic gradient descent algorithm for multiparty classification," in *Int. Conf. on Artificial Intel. and Statistics*, 2012, pp. 933–941.

[17] B. Jayaraman, L. Wang, D. Evans, and Q. Gu, "Distributed learning without distress: Privacy-preserving empirical risk minimization," *Adv. in Neural Inf. Proc. Sys.*, 2018.

[18] Y. Wang, S. Adams, P. Beling, S. Greenspan, S. Rajagopalan, M. Velez-Rojas, S. Mankovski, S. Boker, and D. Brown, "Privacy preserving distributed deep learning and its application in credit card fraud detection," in *IEEE Int. Conf. on Trust, Sec. & Priv. in Comp. and Comm.*, 2018, pp. 1070–1078.

[19] B. K. Beaulieu-Jones, W. Yuan, S. G. Finlayson, and Z. S. Wu, "Privacy-preserving distributed deep learning for clinical data," *arXiv:1812.01484*, 2018.

[20] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. on Inf. Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[21] D. S. Maioli, C. Lavor, and D. S. Gonçalves, "A note on computing the intersection of spheres in $\mathbb{R}^n$," *The ANZIAM Journal*, vol. 59, no. 2, pp. 271–279, 2017.

[22] E. Grosswald, *Representations of integers as sums of squares*, Springer, 1985.

[23] K. Ireland and M. Rosen, *A classical introduction to modern number theory*, vol. 84, Springer, 2013.

[24] Y. LeCun, C. Cortes, and C. J.C. Burges, "MNIST handwritten digit database," *[Online]. http://yann. lecun. com/exdb/mnist*.