Efficient Privacy-preserving Machine Learning in Hierarchical Distributed System

Qi Jia, *Student Member, IEEE,* Linke Guo, *Member, IEEE,* Yuguang Fang, *Fellow, IEEE,* Guirong Wang,

Abstract—With the dramatic growth of data in both amount and scale, distributed machine learning has become an important tool for the massive data to finish the tasks as prediction, classification, etc. However, due to the practical physical constraints and the potention privacy leakage of data, it is infeasible to aggregate raw data from all data owners for the learning purpose. To tackle this problem, the distributed privacy-preserving learning approaches are introduced to learn over all distributed data without exposing the real information. However, existing approaches have limits on the complicated distributed system. On the one hand, traditional privacy-preserving learning approaches rely on heavy cryptographic primitives on training data, in which the learning speed is dramatically slowed down due to the computation overheads. On the other hand, the complicated system architecture becomes a barrier in the practical distributed system. In this paper, we propose an efficient privacy-preserving machine learning scheme for hierarchical distributed systems. We modify and improve the collaborative learning algorithm. The proposed scheme not only reduces the overhead for the learning process but also provides the comprehensive protection for each layer of the hierarchical distributed system. In addition, based on the analysis of the collaborative convergency in different learning groups, we also propose an asynchronous strategy to further improve the learning efficiency of hierarchical distributed system. At the last, extensive experiments on real-world data are implemented to evaluate the privacy, efficacy, and efficiency of our proposed schemes.

Index Terms—Efficiency, Privacy, Hierarchical Distributed System, Machine Learning.

1 Introduction

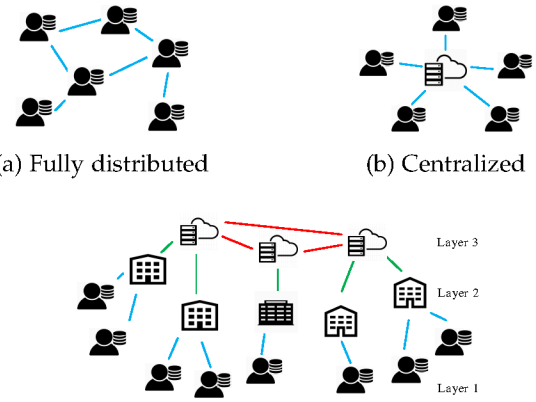
Machine learning is an important data analysis tool in the classification, regression, and prediction tasks for the large-scale dataset. With the development of network technologies, various types of data are increasingly generated and stored in distributed systems [1], [2]. The distributed learning approaches are becoming more desired to discover the critical information from the distributed datasets. Whereas the existing privacy issues are always the stumbling blocks in the development of the distributed machine learning techniques. In most cases, the datasets are possessed by different distributed data owners, and the data owners may be reluctant to expose their data to other parties due to the privacy concerns. As a result, the privacy-preserving distributed machine learning approaches [3], [4], [5], [6], [7], [8], [9] are developed for tackling such privacy issues while achieving the learning at the same time.

However, the conventional privacy-preserving distributed machine learning approaches focus on the simple distributed system architectures, which requires heavy computation loads or can only provide learning schemes over the restricted scenarios. In this paper, we take the step forward to study the more complex hierarchical architecture of the distributed system and propose the corresponding privacy-preserving learning strategies. As illustrated in Fig. 1, the simple distributed systems include fully distributed (Fig. 1(a)) or centralized (Fig. 1(b)) architectures, which only contain single layer with one kind of distributed users. On the contrary, the hierarchical architecture (Fig. 1(c)) includes multiple layers and different layer plays different roles in the whole distributed system. Compared to the simple architectures, the hierarchical distributed system is more common to see in the real-world. For example, the cellular network has such architecture, where the mobile devices, base stations, and switching centers are shaping the different layers in the whole system. Another example could be the cooperations over different companies with their own departments and each department has its own customers' sensitive data. In such hierarchical distributed system, not only the simple distributed users of the lowest layer have the privacy requirements of their data, but also the agents or servers in the upper layers should provide the privacy preservation for the learning process, where the conventional approaches are not suitable anymore.

To tackle the existing problems, we propose an efficient privacy-preserving learning approach over the hierarchical system. Instead of passing the data with randomness or encryption from the traditional privacy learning approach, we decompose the optimization of learning task to the distributed users to avoid the direct data transmission, which can save the additional operations and prevent the exposure of data. Compared to the learning decomposition of simple system architectures, our analysis in the hierarchical system has more complicated privacy requirements of transmitting parameters from the network structure. Therefore, according to the different requirements, we analyze the different data partitions of distributed users and propose the corresponding secure strategies to solve the possible privacy

- *Mr. Jia and Dr. Guo are with the Department of Electrical and Computer Engineering, Binghamton University, Binghamton, NY, 13850. E-mail: qjia1@binghamton.edu, lguo@binghamton.edu*
- *Dr. Fang is with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, 32611. E-mail: fang@ece.ufl.edu*
- *Dr. Wang is with the Department of Surgery, SUNY Upstate Medical University, Syracuse, NY, 13210. E-mail: wangg@upstate.edu*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2018.2859420, IEEE Transactions on Network Science and Engineering



(a) Fully distributed (b) Centralized

(c) Hierarchical

Fig. 1: Different Distributed System Architectures

leakage of the parameters. Meanwhile, due to the special hierarchical architecture, we study the learning relationship of different groups in the system and further propose an asynchronous strategy of different groups to improve the learning efficiency.

Our contributions are summarized as follows:

- As far as we know, we are the first one to address the privacy-preserving learning problem for the hierarchical distributed system, which provides a new secure learning perspective for the complex network structures.
- Based on the analysis of a two-layer hierarchical distributed system, both the horizontal and vertical data partition scenarios are studied to adapt different learning requirements for the real applications.
- We further discuss the learning properties of the different groups of hierarchical architecture. Based on the analysis, we apply the asynchronous learning strategies to improve the system efficiency.
- We conduct extensive simulations and experiments on multiple data sources to evaluate the efficacy and efficiency of our approach.

The remainder of this paper is organized as follows. In Section 2, we briefly review the related works. The necessary preliminaries are listed in Section 3. Then, in Section 4, we present the basic system overview. After that, we elaborate our privacy-preserving approach for horizontally and vertically data partitioned scenarios in Section 5. The further improvement of asynchronous machine learning is introduced in Section 6. In Section 7, we provide the privacy analysis and evaluated the performance of our scheme from different aspects. Finally, we conclude our paper in Section 8.

2 Related Work

The research works of privacy-preserving machine learning over the distributed system can be categorized into two different aspects, explicit or implicit protection for the data.

The explicit privacy preservation of data is straightforward, where protect operations are directly applied to the data itself. Two main different methods are used to achieve the explicit privacy preservation. One method is to utilize the randomization or perturbation [3], [4], [5], [10], [11] to disguise the information of the original dataset. Generally, before the actual data matrix is transmitted to the other parties, a randomized matrix will be added to it such that the receivers cannot retrieve the real data information due to the randomization. Later, on the learner side, a corresponding randomized learning process is designed over the received data to acquire a randomized result. This learning result will be derandomized on the data owner to retrieve the correct model. The other explicit data privacy preservation method is to apply the cryptographic tools [6], [7], [8], [9], [12], [13]. Due to the homomorphic property of modern encryption schemes, the learning process can be conducted on the ciphertext of data with corresponding computations of plaintext. Under this method, the original data needs to be encrypted on the data owner side and the final result should be decrypted for reaching the correct model.

Although the explicit approaches can achieve the data privacy preservation for the distributed data, the additional operations are added on the original learning process and extra information is introduced to the original data, which makes it impractical when applied to the large-scale distributed system. In order to avoid such inefficiency, the implicit approaches are proposed to exploit the optimization procedures to protect the data privacy [14], [15], [16], [17], [18], [19], [20]. Specifically, it takes the advantage of the decomposition property of distributed optimization algorithms to break the whole system leaning target to separated local learning tasks for the distributed users. In this approach, instead of delivering the data between participated data owners, it transmits the local optimized parameters from one data owner to the another. The optimized parameters are collaborated among data owners. By consenesusing to the convergence of these parameters, all data owners will get a same learning result at the end of learning.

However, in the complicated hierarchical distributed system, the parameter transmmition between multiple different parties still has potential privacy leakage risks. The lack of considerations in the parameter privacy makes these conventional approaches fail to be used in such hierarchical distributed system. Our work address such issues and propose an efficient learning approach on the hierarchical distributed system with different data partition scenarios. Also, we involve the asynchronous learning process to further improve learning global efficiency.

3 Preliminaries

Before we expound our privacy-preserving learning approach in the hierarchical distributed system, we briefly provide some preliminaries that will be used to in the following sections. The Alternating Direction Methods of Multiplier approach is the core of the collaborative learning. The secure summation protocol and matrix product protocol will be used to ensure the parameter privacy in our proposed schemes.

3.1 Alternating Direction Method of Multiplier

The alternating direction method of multipliers (ADMM) [21] is a variant of the standard augmented Lagrangian

method that different partial updates are generated for the dual variables. Its classic optimization problem is formulated as follows:

$$\min_{\mathbf{x}} \ f(\mathbf{x}) + g(\mathbf{x}) \tag{1}$$

where $\mathbf{x} \in \mathbf{R}^n$ is the optimization variable, and $f, g : \mathbf{R}^n \rightarrow \mathbf{R}$ are the objective functions. If we introduce a new variable $\mathbf{z}$, an equivalent problem can be constructed which having the following form:

$$\min_{\mathbf{x}} \ f(\mathbf{x}) + g(\mathbf{z}), \quad s.t. \, \mathbf{x} = \mathbf{z}. \tag{2}$$

The optimized result of Eq. (2) should be the same as in the Eq. (1). However, in Eq. (2), it presents an alternative approach to solve $\mathbf{x}$ and $\mathbf{z}$ separately. In this form, we can compute $\mathbf{x}$ or $\mathbf{z}$ while treating the other as fixed values. Furthermore, this problem can be generalized as follows:

$$\min_{\mathbf{x},\mathbf{z}} \ f(\mathbf{x}) + g(\mathbf{z}), \quad s.t. \, \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}. \tag{3}$$

where $\mathbf{A}, \mathbf{B} \in \mathbf{R}^{m \times n}$ are the constraint matrices. To solve this optimization problem, ADMM approach utilizes the dual ascent and the augmented multipliers to separately compute the optimal values for each variable, which consists of the iterations for $\mathbf{x}$ and $\mathbf{z}$ as the following steps:

$$\begin{aligned} \mathbf{x}^{t+1} &:= \underset{\mathbf{x}}{\operatorname{argmin}} \{f(\mathbf{x}) + \frac{\rho}{2}||\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z}^t - \mathbf{c} + \mathbf{u}^t||_2^2\} \\ \mathbf{z}^{t+1} &:= \underset{\mathbf{z}}{\operatorname{argmin}} \{g(\mathbf{z}) + \frac{\rho}{2}||\mathbf{A}\mathbf{x}^{t+1} + \mathbf{B}\mathbf{z} - \mathbf{c} + \mathbf{u}^t||_2^2\} \\ \mathbf{u}^{t+1} &:= \mathbf{u}^t + \mathbf{A}\mathbf{x}^{t+1} + \mathbf{B}\mathbf{z}^{t+1} - \mathbf{c} \end{aligned} \tag{4}$$

where $\mathbf{u}$ is a compensate variable. The variables will update with the growth of iterations. The compensate variable $\mathbf{u}$ will converge to $\mathbf{0}$ and the corresponding variables $\mathbf{x}$ and $\mathbf{z}$ will reach to their optimal values.

3.2 Secure Summation Protocol

Secure summation protocol provide that ways to privately calculate the summation among different users without exposing the actual value of each user. We use the secret sharing [22] as a secure computation approach in our scheme. Bascially, assume there are N users in the system and each of them has a value v_i, it has following steps:

- Each user i in the system generates $N-1$ random numbers $r_{ij} (i \neq j)$ and sends the random numbers to $N-1$ other users respectively.
- Each user i sums over its own generated numbers as $g_i = \sum_{j,i \neq j}^{N} r_{ij}$ and its received numbers as $h_i = \sum_{j,j \neq i}^{N} r_{ji}$.
- After calculating these two values, instead of original value v_i, each user sends $v_i + g_i - h_i$ to other users.
- Finally, receivers can calculate the summation value $u = \sum_i^N (v_i + g_i - h_i) = \sum_i^N (v_i + \sum_{j,i \neq j}^{N} r_{ij} - \sum_{j,j \neq i}^{N} r_{ji}) = \sum_i^N v_i$.

3.3 Secure Matrix Products Protocol

To privately compute the product of two matrices from different parties [23], a secure matrix products protocol is required. Suppose there are two users A and B in the system. $\mathbf{X}^A$ is a $(n \times p_A)$ data matrix of user A and $\mathbf{X}^B$ is a $(n \times p_B)$ data matrix of user B. They wish to compute the products $(\mathbf{X}^A)^T\mathbf{X}^B$ of these two matrices without revealing the actual matrix values to each other. The protocol has following steps:

- User A splits its matrix $\mathbf{X}^A$ into p_A different columns, $\mathbf{X}^A = [\mathbf{X}_1^A \mathbf{X}_2^A ... \mathbf{X}_{p_A}^A]$. Then, user A generates g different n-dimensional vectors $\{\mathbf{Z}_1, \mathbf{Z}_2, ..., \mathbf{Z}_g\}$ to ensure that each vector has dot product 0 with the columns, i.e., $\mathbf{Z}_i^T \mathbf{X}_j^A = 0$ for all i and j. After that, user A sends the $(n \times g)$ dimensional matrix $\mathbf{Z} = [\mathbf{Z}_1 \mathbf{Z}_2 ... \mathbf{Z}_g]$ to user B.
- User B computes $\mathbf{W} = (\mathbf{I} - \mathbf{Z}\mathbf{Z}^T)\mathbf{X}^B$, where $\mathbf{I}$ is an $(n \times n)$-dimensional identity matrix. Then, user B sends $\mathbf{W}$ back to user A.
- Finally, user A calculates $(\mathbf{X}^A)^T\mathbf{W} = (\mathbf{X}^A)^T(\mathbf{I} - \mathbf{Z}\mathbf{Z}^T)\mathbf{X}^B = (\mathbf{X}^A)^T\mathbf{X}^B$ and share the product value $(\mathbf{X}^A)^T\mathbf{X}^B$ to user B.

4 System Overview

In this section, we describe the model of the hierarchical distributed system, introduce the existing privacy challenges, and explain the basic privacy-preserving collaborative learning method.

4.1 Model Description

In the hierarchical distributed system, the entities can be divided into different layers. Without loss of generality, we study a two-layer hierarchical distributed system. The learning of complicated system with more layers can be similarly deduced from the two-layer architecture. As shown in Fig. 2, an example of a two-layer hierarchical distributed system is built by the doctors and hospitals. Basically, the lower layer is consisted by the actual data owners, such as the doctors having their own patient health records (PHRs). The upper layer has multiple nodes to assist different data owners for collecting, transmitting, or computing with other data owners, such as the hospitals or therapy centers can manage different doctors and they can have cooperations with each other. For simplicity, we name the entities in the lower layer as users and the nodes in the upper layer as agents. Generally, the users are separated into different groups with their agents. The users should have the ability to communicate with each other in the same group and the agents can exchange information with other agents. In this example, it means that one doctor can exchange information with other doctors in the same hospital, while one hospital is able to cooperate with other hospitals. The goal is to deploy the machine learning process over the whole system for acquiring a predication model based on all the distributed data, which means a medical diagnosis model can be set up from different hospitals and doctors without sacrifies the privacy of PHRs.

Specifically, we assume that there are totally N users in the lower layer, and they are categorized into S different groups by the agents. Each group i $(1 \leq i \leq S)$ has a group agent to provide services for its N_i group users. Each user j $(1 \leq j \leq N_i)$ of group i has its own dataset as a $(n_{ij} \times p_{ij})$ matrix $\mathbf{X}_{ij}$ and the corresponding $(n_{ij} \times 1)$ label vector $\mathbf{Y}_{ij}$. For a learning scheme $\mathcal{L}$, the goal is to apply it in the system

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2018.2859420, IEEE Transactions on Network Science and Engineering

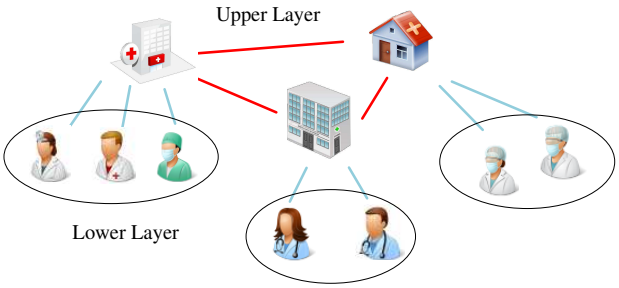


Fig. 2: Two-layer Hierarchical Architecture Model

to find the model $\mathcal{M} : \mathbb{R}^{p+1} \rightarrow \{0, 1\}$ over all the distributed users' data.

4.2 Privacy Challenges

We assume all the entities in the system are honest-but-curious. It means that they will strictly follow the proposed scheme, but they are curious to learn more information than allowed by deducing from what they obtained. We also assume the collusion will not happen between the users and agents, which means the users or agents will not share the unauthorized information to others during the learning process. Besides, we do not consider the outside attacks or the user off-line cases in the distributed system. We only focus on analyzing the possible data privacy leakages from the system inside due to the architecture complexity. Based on such assumptions, there are two major privacy challenges for the distributed users.

First, due to the sensitivity of data, the raw data information of distributed users should not be directly exposed to others in the system. For the user j in group i, neither the rows nor columns of the data matrix $\mathbf{X}_{ij}$ should be realized by any other users or agents. For example, without the consent of patients, any parts of the PHRs cannot be exposed to other entities.

Second, the data should not be retrieved from the obtained information during the learning process. Although the data is not directly exposed, the parameters transmission between different users or agents cannot be prevented. Usually, the transmitting parameters are some optimal values from the distributed data, which include part of the original information. Even it cannot be used to perfectly reconstruct all the original data, it still reflects some basic knowledge of data. For example, in the SVM learning scheme, the optimal values from the local computation of distributed users are the Support Vectors, which still represent the boundary data positions of different classes. Therefore, protection measures are required to prevent such leakage from the parameters transmitting.

4.3 Privacy-preserving Collaborative Learning

Based on above model assumptions and privacy challenges, we reveal the limitation of the basic privacy-preserving collaborative learning approach.

Most machine learning schemes have an objective cost function h to be optimized for finding the best solutions for model parameters. Without loss of generality, the optimization problem can be formulated as a minimization task:

$$\min_{\mathbf{w}} h(\mathbf{w}) \tag{5}$$

where $\mathbf{w}$ are the optimal model parameters. For example, in the linear SVM, the optimization problem is:

$$\begin{aligned} \underset{\mathbf{w},\mathbf{b}}{\operatorname{argmin}} \frac{1}{2} \sum_{SVs} ||\mathbf{w}||^2 + C||\boldsymbol{\xi}||_1^1 \\ s.t. \;\; \mathbf{Y}(\mathbf{X}\mathbf{w} + \mathbf{1}\mathbf{b}) \geq \mathbf{1} - \boldsymbol{\xi}, \\ \boldsymbol{\xi} \geq \mathbf{0}. \end{aligned} \tag{6}$$

where $\mathbf{w}$ and $\mathbf{b}$ are the model parameters. They are used in the decision function $d(\mathbf{t}) = \mathbf{w}^T\mathbf{t} + \mathbf{b}$ for classifying the coming test data $\mathbf{t}$ to different output labels [24]. The soft margin parameter C and the slack variable $\boldsymbol{\xi}$ are used to control the strictness and outliers of the classifier. The matrix $\mathbf{X}$ represents the training dataset and the vector $\mathbf{Y}$ includes the training labels for each data item in the rows of $\mathbf{X}$.

However, in a distributed system, the dataset $\mathbf{X}$ and the label vector $\mathbf{Y}$ are split among the distributed users. Directly collecting the data and labels from users to solve the learning objective function will violate the privacy requirements of user's data. So, we can utilize a new variable to rewrite the optimization problem as follows [14]:

$$\min_{\mathbf{w}_i} h_i(\mathbf{w}_i), \quad s.t. \, \mathbf{w}_i = \mathbf{z} \tag{7}$$

where h_i is the objective function and $\mathbf{w}_i$ is the model parameter for each user i in the distributed system. The newly introduced parameter $\mathbf{z}$ forces all the model parameters $\mathbf{w}_i$ to keep the same result at the end of this optimization. This transformation breaks the optimization problem from the global dataset to the distributed users. Then, according to the ADMM algorithm, we can treat $\mathbf{w}_i$ and $\mathbf{z}$ separately. As a result, each optimization of h_i can be written as follows [15]:

$$\begin{aligned} \mathbf{w}_i^{t+1} &= \underset{\mathbf{w}_i}{\operatorname{argmin}} \{h_i(\mathbf{w}_i) + \frac{\rho}{2}||\mathbf{w}_i - \mathbf{z}^t + \mathbf{r}_i^t||^2\} \\ &= \mathbf{w}_i^t - \alpha \bigtriangledown h_i - \alpha \bigtriangledown \{\frac{\rho}{2}||\mathbf{w}_i - \mathbf{z}^t + \mathbf{r}_i^t||^2\} \end{aligned} \tag{8a}$$

$$\mathbf{z}^{t+1} = \frac{1}{N} \sum_i^N \mathbf{w}_i^{t+1} \tag{8b}$$

$$\mathbf{r}_i^{t+1} = \mathbf{r}_i^t + \mathbf{w}_i^{t+1} - \mathbf{z}^{t+1} \tag{8c}$$

where $\frac{\rho}{2}||\mathbf{w}_i - \mathbf{z}^t + \mathbf{r}_i^t||^2$ is the regularization term, $\bigtriangledown h_i = \frac{\partial h_i}{\partial \mathbf{w}_i}$, and $\bigtriangledown\{\frac{\rho}{2}||\mathbf{w}_i - \mathbf{z}^t + \mathbf{r}_i^t||^2\} = \rho(\mathbf{w}_i - \mathbf{z}^t + \mathbf{r}_i^t)$.

The original global learning optimization problem Eq. (5) is decomposed to iterative updates for distributed users. In each iteration, user i only need to compute the local optimization for its parameter $\mathbf{w}_i$ with its own data from Eq. (8a) and its compensate value $\mathbf{r}_i$ from Eq. (8b). After that, instead of the original data, the local optimized model parameters $\mathbf{w}_i$ will be sent out to other users to calculate the consensus term $\mathbf{z}$ for the next iteration. Along with the growth of iteration, the consensus term $\mathbf{z}$ will become to a stable value. As long as the absolute difference of consensus terms $|\mathbf{z}^t - \mathbf{z}^{t-1}|$ is decreasing to a certain criterion, the iteration will stop and all the users are considered to have a convergent consensus optimization result $\mathbf{z}^t$. This

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2018.2859420, IEEE Transactions on Network Science and Engineering

result will be close to the learning result of gathering all the users' data [21]. Generally, the transmitting model parameters have smaller size and take fewer computation loads for distributed users than transmitting randomized or encrypted data.Therefore, this approach provides a more efficient learning process than the traditional randomness or encryption based methods, while it still keeps the data in private from the direct exposure.

However, two existing issues are the obstructions for applying the basic collaborative learning approach to the hierarchical distributed system. First, the transmitting model parameters $\mathbf{w}_i$ include the hidden information of the original data. To prevent the possible leakage, protections should be provided on the model parameters. Second, the privacy requirements in the hierarchical distributed system are more complicated than the simple architectures. It is necessary to reconsider the learning process and parameter transmission with the system structures.

5 Privacy-preserving Machine Learning for Hierarchical Distributed System

In this section, we present the details of our privacy-preserving learning approach for the hierarchical distributed system. We analyze the parameter transmission in the hierarchical architecture and provide corresponding privacy preserving methods to ensure the secure computation.

Particularly, in the hierarchical distributed system, data partitions make a big difference in the learning process. The horizontally or vertically partitioned data in lower layer users represent different information. Regarding to different data partition scenarios, we provide two different protection strategies to confirm the parameter privacy, which makes our scheme become more practical.

5.1 Horizontally Partitioned Hierarchical System

The distributed user data in horizontally partitioned hierarchical architecture have same data feature dimensions, but they are divided into different groups, where all the users' data can build a complete global dataset matrix. For example, as shown in Fig. 3, at the lower layer, the distributed PHRs of doctors include all the dimensions' information as blood pressure, blood sugar, and blood lipids. At the upper layer, the doctors are assistant by different hospitals and these hospitals play as messengers to pass the data related information. Therefore, we firstly study the basic collaboration in the lower layer, and then figure out what should be transmitted at the higher layer in such system.

5.1.1 Lower Layer Optimization

From the view of the lower layer, users are assisted by the group agent. The learning process can be simplified to a basic collaborative learning scheme. In each iteration, users in the group needs to be guided by the consensus term and regularization term to compute the local optimizations from their own datasets. Specifically, for the group i and its user j, the corresponding iteration updates to Eq. (8a) and Eq, (8c) will be modified as follows:

$$\begin{aligned}\mathbf{w}_{ij}^{t+1} &= \mathbf{w}_{ij}^{t} - \alpha \bigtriangledown h_{ij} - \alpha \bigtriangledown \{\frac{\rho}{2}||\mathbf{w}_{ij} - \mathbf{z}_i^t + \mathbf{r}_{ij}^t||^2\} \\ \mathbf{r}_{ij}^{t+1} &= \mathbf{r}_{ij}^{t} + \mathbf{w}_{ij}^{t+1} - \mathbf{z}_i^{t+1}\end{aligned} \tag{9}$$

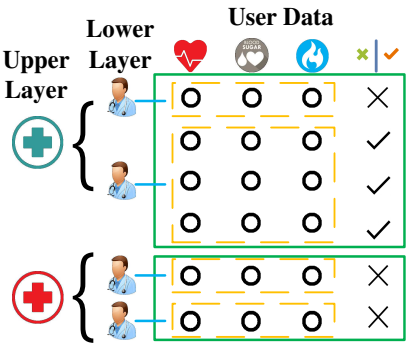


Fig. 3: Horizontally Partitioned Data

These update steps only take the local data from the user j in group i. It does not require the participation of other users' data, where the data privacy is preserved from the direct exposure at the lower layer.

5.1.2 Upper Layer Cooperation

At the upper layer, the parameter transmission is different to the basic scheme. In the fully distributed system, the consensus term $\mathbf{z}$ can be computed from the connection between neighbors. In the centralized distributed system, this consensus term is able to be computed by the collection of all users' model parameters. However, due to the hierarchical architecture, the different groups will have their own consensus term $\mathbf{z}_i$. Then, the original optimization problem needs to be modified as:

$$\min_{\mathbf{w}_{ij}} h_{ij}(\mathbf{w}_{ij}), \quad s.t.\, \mathbf{w}_{ij} = \mathbf{z}_i,\, \mathbf{z}_i = \mathbf{z}. \tag{10}$$

In this problem, the constraints are separated from $\mathbf{z}$ to $\mathbf{z}_i$. Thus, the consensus term $\mathbf{z}$ of Eq. (8b) no longer simply equals to the average of all the distributed users. Instead, each group needs to compute its own consensus term $\mathbf{z}_i$. However, the computation of $\mathbf{z}_i$ not only requires the user's model parameters $\mathbf{w}_{ij}$ from its own group i, but also needs all the consensus terms $\mathbf{z}_j, i \neq j$ from other groups. So, in each iteration of the group agent, it should have the following updates:

$$\begin{aligned}\mathbf{z}_i^{t+1} &= \frac{1}{N}\sum_{i}^{S}\sum_{j}^{N_i}\mathbf{w}_{ij}^{t+1} \\ &= \frac{1}{\sum_k^S N_k}(\sum_{k,k\neq i}^{S} N_k\mathbf{z}_k^t + N_i\mathbf{z}_i^t) \\ &= \frac{1}{\sum_k^S N_k}(\sum_{k,k\neq i}^{S} N_k\mathbf{z}_k^t + \sum_{ij}^{N_i}\mathbf{w}_{ij}^t)\end{aligned} \tag{11}$$

In Eq. (11), different to the original scheme, the group consensus term $\mathbf{z}_i$ becomes an updated element calculated from all the other groups' consensus terms. Such a change of the consensus term makes a significant difference to the learning process. Because it divides the original $\mathbf{z}$ into different groups of $\mathbf{z}_i$, which shrinks the consensus scale from the whole system datasets to several small groups. Then, the negotiation efforts for users' convergence are decreased from global $\mathbf{z}$ to separated $\mathbf{z}_i$, such that the learning speed will

be improved. Therefore, compared to the simple distributed system, the learning of the hierarchical architecture requires fewer iterations to reach the same consensus result, which is shown in our later experiments.

5.1.3 Privacy Preservation

Based on the updating step of Eq. (11), for each group, its agent is in charging of communicating with other groups to cooperatively compute $\mathbf{z}_i$, and it also needs to collect optimized values $\mathbf{w}_{ij}$ for its group users to compute $\sum_{ij}^{N_i} \mathbf{w}_{ij}^t$. These two roles in the hierarchical distributed system have two different privacy requirements.

At the lower layer, the summation of each user's parameter $\sum_j^{N_i} \mathbf{w}_{ij}$ should be obtained by the group agent. To ensure the transmitted user parameters will not be exposed to others, we apply the secret sharing secure summation protocol. First, each user j in the group of i generates $N_i - 1$ random matrices $\mathbf{r}_{jk}(j \neq k)$ and sends the random matrices to other users through their own connections, which means these random matrices are kept unknown to the group agent. Then, each user respectively computes the summation $\mathbf{g}_j$ from generated random matrices, and the summation $\mathbf{h}_j$ of received matrices. After that, each user sends the randomized matrix $\mathbf{w}_{ij} + \mathbf{g}_j - \mathbf{h}_j$ to the group agent. Finally, the group agent collect all the randomized matrices to obtain the summation of all users' parameter by computing $\sum_j^{N_i}(\mathbf{w}_{ij} + \mathbf{g}_j - \mathbf{h}_j) = \sum_j^{N_i} \mathbf{w}_{ij}$.

At the upper layer, where the agents are connected with each other, the update of consensus term $\mathbf{z}_i$ requires all the participations from other groups. Since the consensus term $\mathbf{z}_i$ represents its group data structure, the secure summation for all other consensus terms $\sum_{k,k\neq i}^{S} N_k \mathbf{z}_k$ is still required. To further reduce the computation load, we utilize the a simple secure summation protocol to privately compute this value. In particular, the agent of the group i can first generate a random matrix $\mathbf{r}_i$ which has same dimensions as $\mathbf{z}_i$. Then, this random matrix will pass through every other group agents through the network, where each group agent adds its own consensus term on the received matrix as $\mathbf{r}_k = \sum_{n=i+1}^{k} N_n \mathbf{z}_n + \mathbf{r}_i$. At the end, this matrix returns to the agent of the group i and the summation will be obtained by subtracting the random matrix $\mathbf{r}_i$ as $\sum_{k,k\neq i}^{S} N_k \mathbf{z}_k$. The usage of such secure summation protocol can also make sure the learning process with group agent off-lines. Even one agent becomes to off-line during the learning, as long as it is not the initial one, the summation can still be obtained. The only sacrifice is that the off-line group's data will not be covered in the final learning result.

5.1.4 Application of SVM

To illustrate the feasibility of the proposed algorithm in the specific learning scheme, we use the linear SVM as an example. According to the hierarchical architecture analysis, the linear SVM cost function and its constraints in the horizontally partitioned scenario can be written as following distributed form [14]:

$$\begin{aligned} &\underset{\mathbf{w}_{ij},\mathbf{b}_{ij}}{\operatorname{argmin}} \frac{1}{2}\sum_{i}^{S}\sum_{j}^{N_i} ||\mathbf{w}_{ij}||^2 + C||\boldsymbol{\xi}_{ij}||_1^1 \\ &s.t.\ \ \mathbf{Y}_{ij}(\mathbf{X}_{ij}\mathbf{w}_{ij} + \mathbf{1}\mathbf{b}_{ij}) \geq \mathbf{1} - \boldsymbol{\xi}_{ij}, \\ &\boldsymbol{\xi}_{ij} \geq \mathbf{0},\ \mathbf{w}_{ij} = \mathbf{z}_i, \mathbf{z}_i = \mathbf{z}. \end{aligned} \tag{12}$$

If we consider $\mathbf{v}_{ij} = [\mathbf{w}_{ij}^T, \mathbf{b}_{ij}]^T$, $\mathbf{E}_{ij} = [\mathbf{X}_{ij}^T, \mathbf{1}]^T$. By applying our approach to distributed users, the update iteration step is as follows:

$$\boldsymbol{\lambda}_{ij}^{t+1} = \underset{\boldsymbol{\lambda}:\mathbf{0}\leq\boldsymbol{\lambda}_{ij}\leq\mathbf{C}}{\operatorname{argmax}} \frac{1}{2}\boldsymbol{\lambda}_{ij}^T\mathbf{A}\boldsymbol{\lambda}_{ij} + \mathbf{B}^T\boldsymbol{\lambda}_{ij} \tag{13a}$$

$$\mathbf{v}_{ij}^{t+1} = \frac{N_i}{1+\rho N_i}(\rho\mathbf{z}_i^t - \rho\mathbf{r}_{ij}^t + \boldsymbol{\lambda}_{ij}^{t+1}\mathbf{Y}_{ij}\mathbf{E}_{ij}) \tag{13b}$$

$$\mathbf{z}_i^{t+1} = \frac{1}{N}\sum_{i}^{S}\sum_{j}^{N_i}(\mathbf{v}_{ij}^{t+1} + \mathbf{r}_{ij}^t) \tag{13c}$$

$$= \frac{1}{\sum_k^S N_k}[\sum_{k,k\neq i}^{S} N_k\mathbf{z}_k^t + \sum_{ij}^{N_i}(\mathbf{v}_{ij}^t + \mathbf{r}_{ij}^t)] \tag{13d}$$

$$\mathbf{r}_{ij}^{t+1} = \mathbf{r}_{ij}^t + \mathbf{v}_{ij}^{t+1} - \mathbf{z}_i^{t+1} \tag{13e}$$

where $\boldsymbol{\lambda}_{ij}^{t+1}$ is the optimal Lagrange multiplier at iteration $t+1$ for the user j of group i, and $\mathbf{A} = \frac{N_i}{1+\rho N_i}\mathbf{Y}_{ij}\mathbf{E}_{ij}\mathbf{E}_{ij}^T\mathbf{Y}_{ij}^T$, $\mathbf{B} = -\mathbf{1} + \frac{N_i\rho}{1+\rho N_i}(\mathbf{z}_i^t - \mathbf{r}_{ij}^t)\mathbf{Y}_{ij}\mathbf{E}_{ij}$. In each iteration, the agent of group i applies the different secure summation protocols to compute its own consensus term $\mathbf{z}_i^{t+1}$ from the average value of other groups' consensus terms $\mathbf{z}_{k,k\neq i}^t$, its own users' parameters $\mathbf{v}_{ij}^t$, and its own compensate variable $\mathbf{r}_{ij}^t$. The other updating steps can be locally computed by user j in group i. Note that our approach can also be used to support other machine learning algorithms. Basically, most experience heuristic optimized tasks, not only linear SVM, but also the normal nonlinear SVM, the linear regression, logistic regression, naive Bayesian, and even formulated decision tree, such kind of machine learning approaches can be applied to our approach. The difference will only be the change of objective functions $h(\mathbf{w})$ in Eq. (5).

5.2 Vertically Partitioned Hierarchical System

Different to the horizontally partitioned scenario, the distributed datasets have different data features in the vertically partitioned hierarchical distributed system. As shown in Fig. 4, at the lower layer, different doctors have their own records with same blood feature information. But at the upper layer, the blood information is vertically partitioned, different hospitals may hold different blood features. If we put their data together, they should align with the disease diagnosis results of same patients. As a result, the distributed users can be seen as the mixed segmentations to the global matrix. It vertically divides the matrix so that each data group has different columns at the upper layer, but at the lower layer, each user holds this group's rows as the horizontally partitioned data. Due to such difference, the agents of groups require certain computation at the upper level to assist the consensus of different data features. Therefore, we will explain the learning process at upper layer first, and then give the details of the lower layer cooperations.

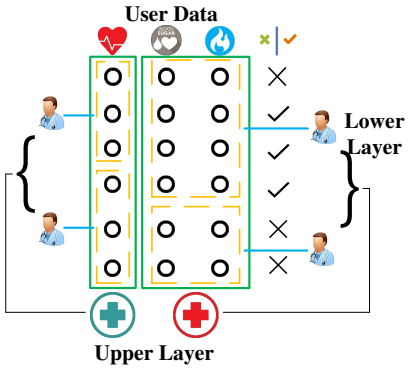


Fig. 4: Vertically Partitioned Data

5.2.1 Upper Layer Optimization

Different to the horizontally partitioned scenario, the analysis for the vertically partitioned data should start from the view of the upper layer. Because the groups have different data features, the responsibility of the consensus term is not simple letting all the parameters stay to the same value. It should integrate all the dimensions parameters and combined value becomes to the real objective function in the original problem. Therefore, generalized objective function for the upper layer is written as follows:

$$\begin{aligned} &\min_{\mathbf{w}_i,\mathbf{z}_i} \sum_i^S f_i(\mathbf{w}_i) + g(\sum_i^S \mathbf{z}_i) \\ &s.t. \quad \mathbf{z}_i = \mathbf{X}_i^T \mathbf{w}_i, \ i = 1, ..., S \end{aligned} \tag{14}$$

where g is the across function combining different dimension of $\mathbf{z}_i$ together to ensure the learning model is set upon all the data dimensions for a consensus learning result, and $\mathbf{X}_i$ is the virtual integrated matrix for the group i. The constraint promises that the product of parameters $\mathbf{w}_i$ and data $\mathbf{X}_i$ commit to one part of the consensus $\mathbf{z}_i$. This problem is hard to be solved in a direct way. So we find its dual function as follows:

$$\begin{aligned} \Gamma &= \inf_{\mathbf{w}_i,\mathbf{z}_i} \{\sum_i^S f_i(\mathbf{w}_i) + g(\sum_i^S \mathbf{z}_i) + \sum_i^S \boldsymbol{\lambda}_i(\mathbf{X}_i^T \mathbf{w}_i - \mathbf{z}_i)\} \\ &= \sum_i^S \inf_{\mathbf{w}_i} \{f_i(\mathbf{w}_i) + \langle \boldsymbol{\lambda}_i \mathbf{X}_i, \mathbf{w}_i \rangle\} + \inf_{\mathbf{z}_i} \{g(\mathbf{z}) + \langle -\boldsymbol{\lambda}_i, \mathbf{z} \rangle\} \\ &= \sum_i^S -f_i^*(-\boldsymbol{\lambda}_i \mathbf{X}_i) - g^*(\boldsymbol{\lambda}_i) \end{aligned} \tag{15}$$

where $\mathbf{z} = \sum_i^S \mathbf{z}_i$ and $\{f^*, g^*\}$ are the conjugate function of $\{f, g\}$. The Γ is a convex function and only when $\boldsymbol{\lambda}_i = \boldsymbol{\lambda}$ it can have infinite value. If we define the function $\phi_i(\boldsymbol{\lambda}_i) = -f_i^*(-\boldsymbol{\lambda}_i \mathbf{X}_i) - \frac{1}{S} g^*(\boldsymbol{\lambda}_i)$, the original vertically data partitioned problem can be transformed to a horizontal problem:

$$\min_{\lambda_i} \ \phi_i(\boldsymbol{\lambda}_i), \quad s.t. \quad \boldsymbol{\lambda}_i = \boldsymbol{\lambda}. \tag{16}$$

Then, it can be solved by the similar procedures of horizontally partitioned scenario as Eq. (9) and Eq. (11).

5.2.2 Lower layer Cooperation

However, the computations of Eq. (15) are only suitable for the upper layer, and all the conducts are only related to the group agent i. Since we have the horizontally partitioned data at the lower layer, the matrix $\mathbf{X}_i$ is distributed to several users j in this group. Therefore, the actual matrices are the destructions of $\mathbf{X}_i$:

$$\mathbf{X}_i = \begin{bmatrix} \mathbf{X}_{i1} \\ ... \\ \mathbf{X}_{ij} \\ ... \\ \mathbf{X}_{iN_i} \end{bmatrix} \tag{17}$$

where $\mathbf{X}_{ij}$ represents the data submatrix possessed by the user i of the group j, which has same column number but different rows to the $\mathbf{X}_i$. Due to such distributions at the lower layer, the privacy should be considered to prevent the leakage of each user's data. Specifically, the computation which related to the $\mathbf{X}_i$ is not available to directly apply the $\mathbf{X}_i$ into the calculation anymore. A private computation method from the submatrices $\mathbf{X}_{ij}$ should be provided. For example, in the Eq. (14) and Eq. (15), the computation $\mathbf{X}_i^T \mathbf{w}_i$ and $-\boldsymbol{\lambda}_i \mathbf{X}_i$ are related to the data matrix $\mathbf{X}_i$ and they are calculated between the group agent and its own users, where each user's data matrix $\mathbf{X}_{ij}$ should be protected.

5.2.3 Privacy Preservation

Without loss of generality, we use matrix $\mathbf{C}_i$ to represent the multiplier of matrix $\mathbf{X}_i$. The matrix $\mathbf{X}_i$ has N_i users with their submatrices $\mathbf{X}_{ij}$, such that $\mathbf{X}_i = [\mathbf{X}_{i1}^T, ..., \mathbf{X}_{iN_i}^T]^T$. Then, we correspondingly separate the $\mathbf{C}_i$ as $\mathbf{C}_i = [\mathbf{C}_{i1}^T, ..., \mathbf{C}_{iN_i}^T]^T$. The product can be written as follow:

$$\begin{aligned} \mathbf{X}_i^T \mathbf{C}_i &= [\mathbf{X}_{i1}^T, ..., \mathbf{X}_{iN_i}^T] \begin{bmatrix} \mathbf{C}_{i1} \\ ... \\ \mathbf{C}_{iN_i} \end{bmatrix} \\ &= \mathbf{X}_{i1}^T \mathbf{C}_{i1} + ... + \mathbf{X}_{iN_i}^T \mathbf{C}_{iN_i} \end{aligned} \tag{18}$$

The product of original matrix is decomposed to the products of the submatrices. Then, we implement the secure matrix product protocol on the submatrices to privately compute this value. The implementation will be taken place between any user or agent. First, user j generates the compensated matrix $\{\mathbf{Z}_{i1}, ..., \mathbf{Z}_{iN_i}\}$ for $\{\mathbf{X}_{i1}, ..., \mathbf{X}_{iN_i}\}$ respectively. Second, the generated $\mathbf{Z}_{ij}$ is send to the agent for computing $\mathbf{W}_{ij} = (\mathbf{I} - \mathbf{Z}_{ij}\mathbf{Z}_{ij}^T)\mathbf{C}_{ij}$. Third, $\mathbf{W}_{ij}$ is sent back and the $\mathbf{X}_{ij}^T \mathbf{C}_{ij}$ is acquired from computing $\mathbf{X}_{ij}^T \mathbf{W}_{ij} = \mathbf{X}_{ij}^T \mathbf{C}_{ij}$. Finally, the $\mathbf{X}_i^T \mathbf{C}_i$ can be calculated by accordingly adding up the $\mathbf{X}_{ij}^T \mathbf{C}_{ij}$ from Eq. (18) using the secret sharing secure summation protocol between the group agent and users. Once we promise the privacy of matrix product, the data privacy can be preserved during the whole vertically partitioned learning process since there is no other information exchange for lower layer users.

However, this approach cannot guarantee the ideal privacy preservation on all matrix computations [23]. To decrease the possible breaches in computation between user A and B, we can choose an optimal value $g = \frac{p_A}{p_A + p_B} n$ for the user A, so that the generated matrix $\mathbf{Z}$ has minimized mutual information of A and B. Without the collusions of different users, this revealed mutual information is insufficient to recover the original matrix value. In some extreme

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2018.2859420, IEEE Transactions on Network Science and Engineering

cases, the leakage is inevitable for specific matrix values. For example, only one non-zero value in a column of the matrix $\mathbf{I}-\mathbf{Z}\mathbf{Z}^T$ can let user A realize one row for matrix $\mathbf{X}^B$. While the leakage can only retrieve part of rows in the original matrix, and it can be prevented by detecting these special cases in the first place.

5.2.4 Application of SVM

Similarity, we provide an example of linear SVM. The updates process for vertically partitioned data can be concluded as follows:

$$\mathbf{w}_i^{t+1} = \rho(\mathbf{1}+\rho\mathbf{X}_i^T\mathbf{X}_i)^{-1}\mathbf{X}_i^T[\mathbf{z}^t - \sum_i^S \mathbf{X}_i\mathbf{w}_i^t + \mathbf{X}_i\mathbf{w}_i^t + \mathbf{r}^t] \tag{19a}$$

$$\boldsymbol{\lambda}^{t+1} = \underset{\boldsymbol{\lambda}:\mathbf{0}\le\boldsymbol{\lambda}\le\mathbf{C}}{\operatorname{argmax}} \frac{1}{2}\boldsymbol{\lambda}^T\mathbf{A}\boldsymbol{\lambda} + \mathbf{B}^T\boldsymbol{\lambda} \tag{19b}$$

$$\mathbf{z}^{t+1} = \sum_i^S \mathbf{X}_i\mathbf{w}_i^{t+1} - \mathbf{r}^t + \frac{1}{\rho}\boldsymbol{\lambda}^{t+1}\mathbf{Y} \tag{19c}$$

$$\mathbf{r}^{t+1} = \mathbf{r}^t + \mathbf{z}^{t+1} - \sum_i^S \mathbf{X}_i\mathbf{w}_i^{t+1} \tag{19d}$$

where $\mathbf{Y}$ is the labels matrix for the whole system, $\mathbf{z}$ and $\mathbf{b}$ are the parameters of virtual classifier for dividing the hyperplane. $\mathbf{A} = \frac{1}{\rho}\mathbf{Y}\mathbf{1}\mathbf{1}^T\mathbf{Y}$ and $\mathbf{B} = -\mathbf{1}+\mathbf{Y}(\sum_i^S \mathbf{X}_i\mathbf{w}_i^{t+1} - \mathbf{r}^t)$. To provide the privacy preservation for individual users in each group, the secure procedures will be applied to the computation related to the matrix $\mathbf{X}_i$.

6 Asynchronous Learning for Hierarchical Distributed System

6.1 Convergence Analysis

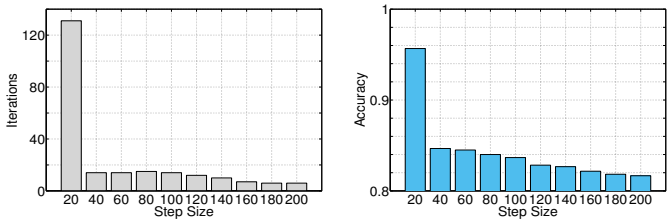
The convergence iteration process of optimization is the key to the privacy-preserving collaborative approach. On the one hand, it prevents the direct data exposure and provides secure exchange of optimized parameters in each iteration. On the other hand, the learning speed performance depends on the collaborative convergence process. For different datasets and system settings, the required iterations to reach the consensus learning result are quite different. Such difference of iterations provides the flexibility of learning in the hierarchical distributed system. We can adjust the settings of the learning to change the convergence speed for different groups in the hierarchical architecture to improve the global learning efficiency.

From the iteration update of parameter $\mathbf{w}$ in the learning process:

$$\mathbf{w}_i^{t+1} = \underset{\mathbf{w}_i}{\operatorname{argmin}} \{h_i(\mathbf{w}_i) + \frac{\rho}{2}||\mathbf{w}_i - \mathbf{z}^t + \mathbf{r}_i^t||^2\},$$

we can see that the convergence speed is influenced by three factors. First, it depends on the dataset itself. Different datasets have their own data structures, which will have different result to $h_i(\mathbf{w}_i)$. If the content of one dataset is closer to the universal data structure, it does not need many iterations for converging to the consensus, and vice versa. Second, the collaborative convergence depends on the influence of other users' data, which is weighted by the parameter ρ in $\frac{\rho}{2}||\mathbf{w}_i - \mathbf{z}^t + \mathbf{r}_i^t||^2$. If the weight of this influence becomes larger, it is faster to reach the convergence result for users since the exchanged information from other users is increased. Third, the convergence also depends on the parameter α in $\alpha \bigtriangledown h_i + \alpha \bigtriangledown \{\frac{\rho}{2}||\mathbf{w}_i - \mathbf{z}^t + \mathbf{r}_i^t||^2\}$, which controls the gradient descent step size. If it grows larger, the increased step size leads to faster convergence but could cause the unsatisfied results. Based on different parameter settings and the data contents, the convergence speed could vary from one to another. At the same time, such difference also influences the performance of the learned result.

To illustrate the influence, we build a simulation on the centralized model for 20 distributed users by different parameter settings in the convergence. As shown in Fig. 5, with the growth of step size, the required iteration number is decreasing. Meanwhile, the accuracy of the learned result has an obvious decline. A trade-off between the efficiency and result performance can be made by adjusting the learning parameter settings. Normally, we need to choose the proper parameters to pursue a balanced learning performance for suiting different applications in practice [25].



(a) Iterations (b) Accuracy

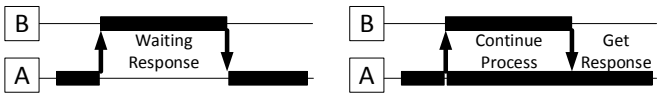
Fig. 5: Different Step Size

6.2 Asynchronous Strategy

In accordance with such convergence facts, an asynchronous learning strategy becomes available in the hierarchical distributed system [26], [27]. Compared to the synchronous one, the asynchronous learning brings two benefits to the system. First, the hierarchical architecture can promise the flexible choice of the different learning parameters for groups in the system. Especially, the same group is more often to have similar user data structures. For example, one hospital may have its local patients' health records and the symptoms could be similar to each other for the patients in this area. Due to such similarity, we can accordingly select the appropriate convergence speed parameters for different groups without introducing the conflicts. Second, the asynchronous learning saves computation time for the convergence. As shown in Fig. 6, different to the synchronous mode, the distributed users with asynchronous learning strategy has no necessity to wait for the response from other users. In the hierarchical distributed system, it means that the different groups can work on their own convergence progress while do not need to wait for other groups consensus results.

Based on these benefits, we can arrange different parameters to the groups. Specifically, in Eq. (8), we change the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2018.2859420, IEEE Transactions on Network Science and Engineering



(a) Synchronous (b) Asynchronous

Fig. 6: Different Synchronization Mode

parameter α, ρ to α_i, ρ_i to represent the different settings for the group i.

In the asynchronous learning, different parameters will lead to the different learning progress of groups. Some groups may have faster convergence to reach the final consensus result, but the others may become slower. However, to ensure the global consensus convergence result, the synchronization is needed for the groups to communicate and exchange their own optimized values. Generally, two methods can be used to keep the synchronization for different groups. One is to have the prearranged learning parameters for different groups so that the synchronization will be operated after several specific iterations. Unfortunately, without knowing the parameter settings from other groups, it can hardly determine the convergence parameters in advance. Also, even the speed parameters can be prearranged, the faster groups still need to wait for the slower ones for synchronization.

To avoid the inefficiency and promise the convergence, we apply modified asynchronous iterations for the hierarchical model. In particular, we do not limit or prearrange the speed parameters of different groups. Instead, we define a staleness value D for the whole network. This staleness value is an upper bound for the distance of fastest group to the slowest one. Although each group has its own convergence speed, this staleness promises the fastest one should not be too much ahead than the slowest one [28], [29]. We can control this staleness so that the system still consensus to the correct solution. As a result, using t_i as the index for the iterations of the group i, our algorithm for the asynchronous distributed privacy-preserving learning in the horizontally partitioned model can be written as following formulas:

$$\begin{aligned}\mathbf{w}_{ij}^{t_i+1} &= \mathbf{w}_{ij}^{t_i} - \alpha_i \bigtriangledown h_{ij} - \alpha_i\rho_i(\mathbf{w}_{ij} - \mathbf{z}_i^{t_i} + \mathbf{r}_{ij}^{t_i}) \\ \mathbf{z}_i^{t_i+1} &= \frac{1}{N}(\sum_{k,k\neq i}^{S} N_k\mathbf{z}_k^{t_k} + \sum_{ij}^{N_i}\mathbf{w}_{ij}^{t_i+1}) \\ \mathbf{r}_{ij}^{t_i+1} &= \mathbf{r}_{ij}^{t_i} + \mathbf{w}_{ij}^{t_i+1} - \mathbf{z}_i^{t_i+1}\end{aligned} \tag{20}$$

in which $\max(t_i) - \min(t_i) \leq D, i \in S$. Since the vertically data partitioned scenario can be transformed to the horizontal problem, the similar asynchronous iterations can be applied to improve the efficiency as well [30].

Comparing to the original learning in Eq. (9) and Eq.(11), the asynchronous process has difference in the gathering of other groups' consensus terms $\sum_{k,k\neq i}^{S} N_k\mathbf{z}_k^{t_k}$. To analyze the influence of this difference. We can firstly study a two users case. Assume there are two distributed users A and B. If we deploy the asynchronous learning process on their data, and we let A be the faster one. We will have $\mathbf{w}_A^1 = \mathbf{w}_A^0 - f(\mathbf{z}_A^0)$, where $f(\mathbf{z}_A^0) = \alpha_A \bigtriangledown h_A + \alpha_A\rho_A(\mathbf{w}_A - \mathbf{z}_A^0 + \mathbf{r}_A^0)$. Then, we will have $\mathbf{z}_A^1 = \frac{1}{2}(\mathbf{z}_B^0 + \mathbf{w}_A^1) = \frac{1}{2}[\mathbf{w}_B^0 + \mathbf{w}_A^0 - f(\mathbf{z}_A^0)]$, and $\mathbf{w}_A^2 = \mathbf{w}_A^1 - f(\mathbf{z}_A^1) = \mathbf{w}_A^0 - f(\mathbf{z}_A^0) - f(\mathbf{z}_A^1)$. So, since we have a largest staleness value D, in the extreme case, we will have $\mathbf{z}_A^D = \frac{1}{2}[\mathbf{w}_B^0 + \mathbf{w}_A^0 - \sum_{i=0}^{D-1} f(\mathbf{z}_A^i)]$. Similarly, in the original learning process, we have $\mathbf{z'}_A^D = \frac{1}{2}[\mathbf{w}_B^0 - \sum_{i=0}^{D-1} f(\mathbf{z'}_B^i) + \mathbf{w}_A^0 - \sum_{i=0}^{D-1} f(\mathbf{z'}_A^i)]$. Because we will not have too large staleness value D, we can assume that $f(\mathbf{z}_A^i) \approx f(\mathbf{z'}_A^i)$. Then, we will have the difference between original and asynchronous learning as follow,

$$\begin{aligned}\mathbf{z'}_A^D - \mathbf{z}_A^D &= \frac{1}{2}[\mathbf{w}_B^0 - \sum_{i=0}^{D-1} f(\mathbf{z'}_B^i) + \mathbf{w}_A^0 - \sum_{i=0}^{D-1} f(\mathbf{z'}_A^i)] \\ &- \frac{1}{2}[\mathbf{w}_B^0 + \mathbf{w}_A^0 - \sum_{i=0}^{D-1} f(\mathbf{z}_A^i)] = -\frac{1}{2}\sum_{i=0}^{D-1} f(\mathbf{z'}_B^i)\end{aligned} \tag{21}$$

Assume it takes M steps in total for the original learning process to reach the final consensus result $\mathbf{z'}_A^m$. Then, the portion of this difference in the original convergence result will be:

$$\frac{\mathbf{z'}_A^D - \mathbf{z}_A^D}{\mathbf{z'}_A^m} = \frac{\sum_{i=0}^{D-1} f(\mathbf{z'}_B^i)}{\sum_{i=0}^{m-1} f(\mathbf{z'}_B^i) - \mathbf{w}_B^0 + \sum_{i=0}^{m-1} f(\mathbf{z'}_A^i) - \mathbf{w}_A^0}, \tag{22}$$

Usually, compared to the whole iteration steps m, D will be a relative smaller value in the learning preset. Also, since this is the extreme case which has largest iteration gap from the beginning step, it only provides an upper bound for the learning difference between original and asynchronous strategies. For the multiple groups in our system, the influence will changed from $-\frac{1}{2}\sum_{i=0}^{D-1} f(\mathbf{z'}_B^i)$ to $-\frac{1}{N-1}\sum_{k\in S,k\neq A}\sum_{i=0}^{D-1} f(\mathbf{z'}_k^i)$, which will not make larger influence for group A. As a consequence, compared to the original learning process, the asynchronous approach will have similar result in the hierarchical distributed system, which means it has the same learning ability, but is more efficient in practice.

7 Performance Evaluation

7.1 Privacy Analysis

In our scheme, the decomposition from Eq. (5) to Eq. (8) disperses the learning process from a united global task to the distributed users. So, for each user, the related computations have three components for $\mathbf{w}_i$, $\mathbf{z}$, and $\mathbf{r}_i$. The direct exposure of each user's data is prevented from the provided locality in the collaborative learning process. Specifically, the model parameter $\mathbf{w}_i$ and compensate parameter $\mathbf{r}_i$ only requires user's own data and an outside consensus term $\mathbf{z}$. The only possible leakage is the computation of consensus term $\mathbf{z}$ from Eq. (8b), which is a summation of model parameters $\mathbf{w}_i$ from different users.

To prevent the possible leakages, our scheme provides the protection with the hierarchical architecture analysis in different applications. In the horizontally data partitioned scenario, the group agents are in charging of the collection and transmission for the model parameter $\mathbf{w}_{ij}$ and group consensus term $\mathbf{z}_i$. At the lower layer, the secret sharing secure summation protocol can prevent the group agent to realize the value of $\mathbf{w}_{ij}$. Even the group agent has all the $\mathbf{w}_{ij} + \mathbf{g}_j - \mathbf{h}_j$, it cannot recover the real $\mathbf{w}_{ij}$ without knowing the randomized matrices $\mathbf{r}_{jk}$. At the upper layer, the simple secure summation protocol can still privately obtain the

consensus term $\mathbf{z}_i$ but has more efficient computation. However, in some extreme cases, it is possible to have leakage of consensus term $\mathbf{z}_i$. For example, if some agents are isolated in the upper layer network, the summation matrix passes to such agents will return back the previous ones. In this case, the previous agent i can realize the consensus term $\mathbf{z}_{i+1}$ by subtracting two neighbor matrices. Therefore, to prevent such issues, the isolated agent can also generate its own random matrix for disguising the summation. At the end, the summation value will be sent to this agent for subtracting its generated random matrix to obtained the real result.

In the vertical data partition scenario, the upper layer actually has a similar privacy-preserving collaborative learning process, where the data privacy of each group agent can be preserved from the locality. Within each group, the secure matrix protocol is applied in the computations of parameters and user's data. The agent cannot reveal the real data matrix value of its user. Besides, since each user only computes its own data with the parameter submatrix in corresponding rows, the data will not be exposed to the other users in the same group. Thus, the data privacy is preserved at both upper and lower layer.

7.2 Analysis of Simple Architectures

Due to the complicated overheads on the training data, traditional randomness or encryption based privacy-preserving learning needs more computational time than the collaborative learning approaches. For example, we tested the encryption-based privacy-preserving learning approach [3] for 20 distributed users on the Higgs dataset [31]. As shown in Table 1, the encryption based learning approach requires much more time cost on the learning process. With the growth of data instance number in each distributed user, it even has more increment than the collaborative learning approach. The related comparison of efficiency also has been illustrated in many previous works [14], [15], [16]. Therefore, in our experiment, we mainly focus on the performance comparison between the basic collaborative learning approach and our proposed schemes.

TABLE 1: Comparison between Encryption-based and Collaborative Distributed Learning

Data Instances	Encryption Based	Fully Distributed
1,00	163.1s	6.8s
1,000	2041.3s	144.2s
5,000	9084.3s	583.1s
10,000	30231.4s	1473.9s

We first present the basic privacy-preserving collaborative learning in two simple architectures, fully distributed and centralized. In the fully distributed architecture [14], one user cannot communicate with all the other users at one single communication step. Instead, each user can only communicate and exchange information with its neighbors. In such case, the $\mathbf{z}^{t+1} = \frac{1}{N_i}\sum_j^{N_i}\mathbf{w}_j^{t+1}$, where N_i is the number of neighbors for user i. Each user cannot collect all $\mathbf{w}_i$ from all other users but only the neighbors' model parameters $\mathbf{w}_i$. Even though, by gradually passing the parameters along with the network, it still promises that the whole network can achieve the convergence [14]. However, due to the insufficient information exchange, the fully distributed architecture needs more iterations to reach the stop criterion of the convergence. On the contrary, the centralized architecture has a central node to connect all the nodes in the network [16]. The central node is not trustworthy and the data should not be directly collected on this node. However, with the help of the node, this architecture equals to a virtual complete connected network for the users, where all users can communicate with each other in one step. As a result, each user can obtain all the other users' optimized model parameters $\mathbf{w}_i$ within one communication round. The communication cost and needed iterations will be reduced.

We conduct a simulation for these two kinds of simple distributed architectures and apply the linear SVM learning scheme on 100 users to illustrate their performance difference. Each user's dataset includes 100 data instance with random labels and patterns with 25 data features. For simplicity, in the fully distributed model, the users are assumed to be connected one-by-one as a chain. The results are shown in TABLE 2. From this table, we can see that the learning of fully distributed system requires more iterations, time cost, and communication than the centralized one. This demonstrate that the sufficient communication in centralized architecture can accelerate the convergence speed for the iterations.

TABLE 2: Comparison of Two Simple Architectures

	Fully Distributed	Centralized
Iteration	514	67
Time Cost	766s	107s
Communication Cost	144KB	10KB
Step Size	5	5
Avg. Neighbors	1.98	1 (real) / 99 (virtual)

7.3 Classification Accuracy

To study the performance of our proposed approach in the hierarchical distributed system, we implement the linear support vector machine learning method over the three real-world datasets. The breast cancer data from [32] has 683 instances and 10 feature attributes, the svmguide1 [33] data has 3,089 instances and 4 different data features and the Higgs from [31] has 11,000,000 data items (we use 100,000) with 28 features. All datasets only have positive or negative labels. About 70% data in each dataset are used to train the linear classifiers and the rest 30% data are tested with the trained classifiers to measure the classification accuracy.

We first evaluate the classification accuracy of the distributed learning approaches in three different architectures. The regular linear SVM learning approach is also evaluated as a benchmark of the comparison, where its training data is not separated. We set 20 distributed users in the system, and we divide the users into two groups with similar user numbers to analyze the proposed hierarchical architecture. The classification accuracy results are shown in TABLE 3, where the distributed learning approaches have the same convergence parameters and stop criterion.

From the table, the normal linear SVM has the best performance for all the datasets since it does not have any lost and obstacle from the distributed networks. For the distributed systems, the learning result of the fully

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2018.2859420, IEEE Transactions on Network Science and Engineering

TABLE 3: Classification Accuracy

Dataset	Breast Cancer	Svmguide1	Higgs
Normal LSVM	96.83%	95.4%	64.4%
Fully distributed	89.67%	84.20%	53.7%
Centralized	95.50%	92.2%	61.16%
Hierarchical (2 Groups)	94.83%	89.70%	57.76%

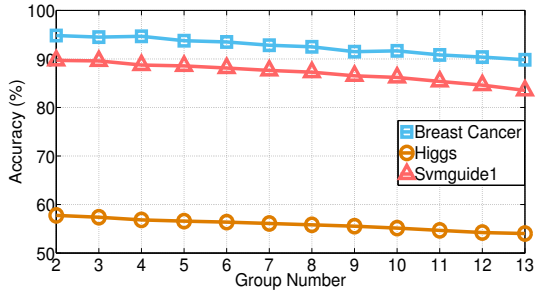


Fig. 7: Classification Accuracy with Different Group Number

distributed architecture has the worst classification accuracy due to the insufficient communications between the users, while the centralized architecture has the best performance. For the hierarchical architecture, its communication is more sufficient than the fully distributed one, but not as good as the centralized architecture. Therefore, it has a balanced performance between these two architectures on the classification accuracy. To analyze the effects of group number for the hierarchical architecture, we extend the group number from 2 to 13 in evaluating the classification accuracy and the results are shown in Fig. 7. With the growth of group number, the accuracy of the hierarchical distributed system is decreasing. Larger group number means the system has a closer architecture to the fully distributed system, which has worse performance on the classification.

7.4 Learning Efficiency

7.4.1 Horizontally Partitioned

To compare the learning cost of the different architectures, we evaluate 4 to 20 distributed users to see the trends. As shown in Fig. 8, the three architectures have the difference on the learning convergence speed. The fully distributed architecture needs the most iterations for reaching the consensus result. Also, with the growing of the user number, the required iterations have an increasing trend. The centralized architecture requires much fewer iterations for the convergence, and it does not have obvious increment with the growth of user number. In particular, since the hierarchical scheme reduces the collaborative range from the whole system to the multiple small groups, it has the best performance with the least iteration requirement. Also, since each group is influenced by the other groups. Once the convergence of one group is stopped, the other groups will quickly align with this converged group, such that the required iterations will be similar in different groups.

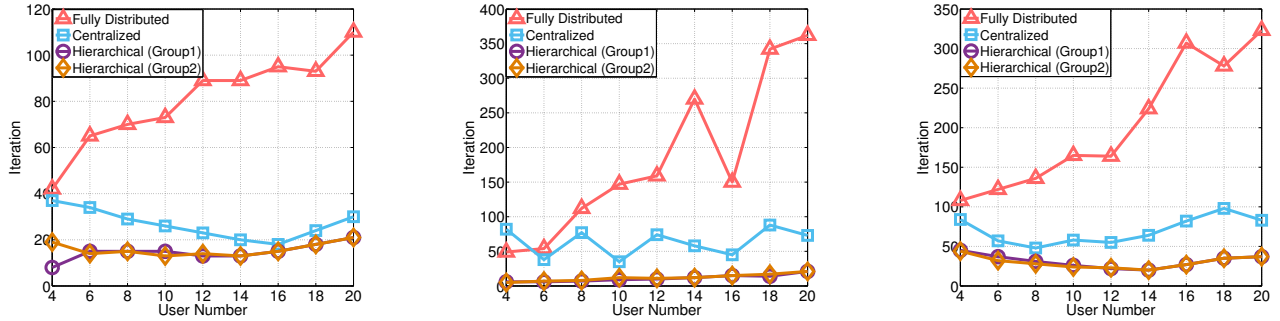
Besides, we test the convergence vibration of different distribution architectures in Fig. 9, which reflects the consensus difference value $|\mathbf{z}^t - \mathbf{z}^{t-1}|$ trends with the growth of iteration. The test is based on all the datasets for 20 distributed users. The y-axis of these figures is the logarithmic value of consensus difference in two adjacent iterations. The longer curve means that it takes more iteration steps to reach the convergence, and the vibrations mean the consensus difference value is changing with the increase of iteration for adjusting the final results. We can see that the curve of fully distributed architecture has a long time of iterations to reach the end of convergence and has fierce vibration tail. It means that the users in the fully distributed system need longer and erratic negotiation period to get a satisfied consensus result at the end of the convergence. The hierarchical and centralized architectures have short curves and reach to the consensus result quickly, which means they do not need the redundant adjustments at the convergence end. Besides, compared to the centralized architecture, the hierarchical distributed system does not have the tail vibrations, which reaches to the consensus result in shorter iterations and it saves more time from the final adjustments between the distributed users.

Moreover, for the hierarchical distributed system, the division of different groups may have influence on the learning performance. We choose different number of groups for the breast cancer and svmguide1 datasets. As shown in Fig. 10(a), we divide the 20 distributed users into 2 to 13 groups in the system, where each group has similar user numbers and the equal data volumes. The results reflect that the different group numbers have impacts on the convergence of the learning. If the group number is larger, the model is getting closer to the fully distributed model. Then, the required number of iteration is increasing. Moreover, the data allocation ratio among the groups also has the impacts. We divide the 20 users into a two-group case of breast cancer data with different ratios. As shown in Fig. 10(b), if the ratio deviation between two groups is larger, it means the data have the larger difference between groups, such that they need different iterations to reach the convergence. However, the influence of this ratio only works for the small size of data in the groups. More data points in each user will mitigate the influence of such deviation. Because the fewer users will have more different typical data structures. With the growing of user number, the effects made by group division ratio will be mitigated. Therefore, combining with the classification accuracy results, the smaller group number and the larger user dataset can result in a better performance in the hierarchical distributed system.

7.4.2 Vertically Partitioned

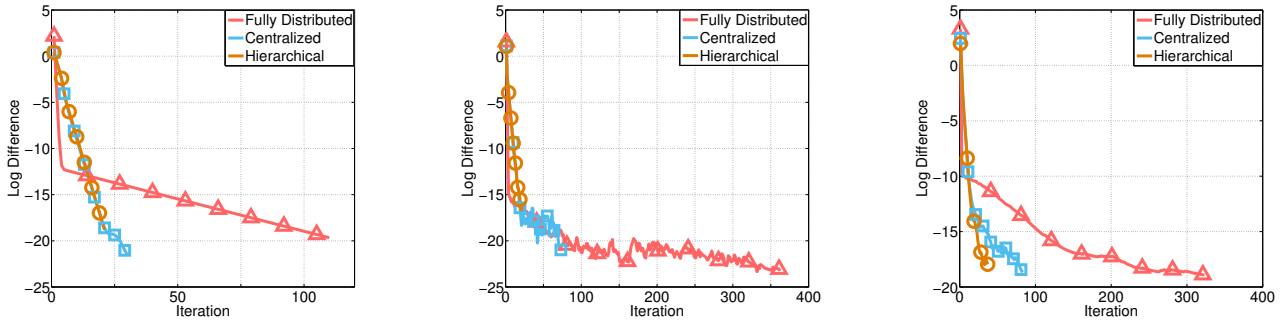
Comparing with the horizontally partitioned one, the major difference happens at the lower layer from the extra matrix computations in the vertically partitioned scenario. So, instead of analyzing the iteration convergence, we evaluate the additional computation load by these secure matrix products. We apply the vertical partition learning approach to the Higgs dataset with three groups of $\{10, 10, 8\}$ data dimensions. The users in each group have their own datasets of different data instance numbers. As shown in Fig. 11, the computational time cost of our privacy-preserving approach introduces a little computation overhead than the original learning scheme, which only comes from the addition secure matrix products.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2018.2859420, IEEE Transactions on Network Science and Engineering



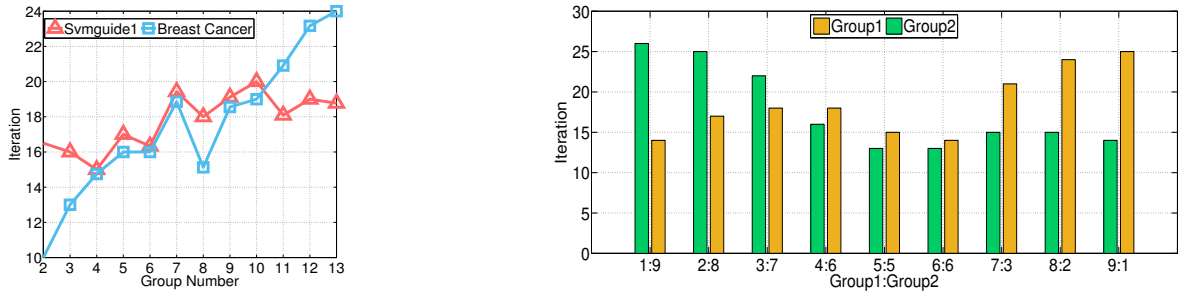
(a) Breast Cancer Data (b) Svmguide1 Data (c) Higgs Data

Fig. 8: Convergence Iteration



(a) Breast Cancer Data (b) Svmguide1 Data (c) Higgs Data

Fig. 9: Convergence Vibration



(a) Different Group Numbers (b) Different Group Ratios

Fig. 10: Group Division

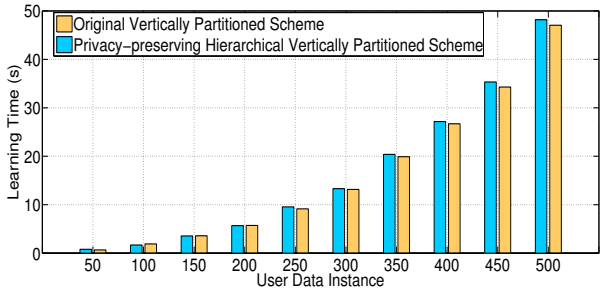


Fig. 11: Hierarchical Vertically Partitioned Scheme

7.5 Improvement of Asynchronous Iterations

Based on the normal hierarchical learning scheme, we further implement the asynchronous learning strategy to the system. The influence of the convergence speed control parameters has been illustrated in Fig. 5, but it is only evaluated with the same step size for different groups in the horizontally partitioned scenario. To study the improvements of asynchronous learning, we evaluate a two-group division to the breast cancer dataset. To see the influence of different learning speeds, the step sizes in different groups are changed. As shown in Fig. 12, the larger step size settings require less iterations for convergence. However, at the same time, the larger step size may lead to the worse classification performance. Due to such difference, we need to choose the balanced step sizes in different groups to satisfy the classification accuracy requirements, while providing a more efficient learning process. For example, if our selection of the step size should not be larger than 50, instead of

using the same $(50, 50)$ in the synchronous iterations, we can choose $(50, 20)$ as the best learning parameters in two groups for the asynchronous learning.

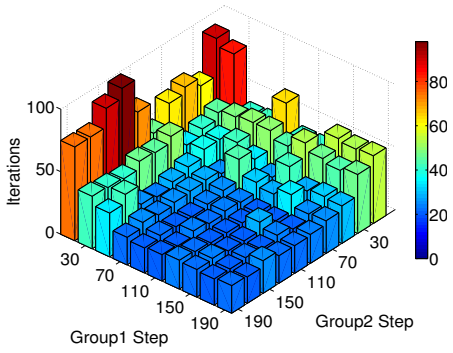


Fig. 12: Different Step Size of Asynchronous Learning

Certainly, this asynchronous learning strategy can also be applied to more groups case in the hierarchical distributed system. As shown in Fig. 13, with different group numbers, the required iterations are evaluated for synchronous and asynchronous strategies. From this figure, the asynchronous learning requires about only half iterations to the synchronous one on the efficiency improvement of the collaborative learning process in the hierarchical distributed system. During the evaluation, we ensure that at least one group in the asynchronous iterations mode has same step size with the synchronous one, such that they are in the same step size level.

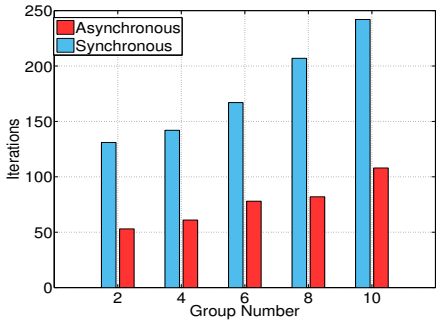


Fig. 13: Iterations in Different Strategies

8 Conclusion

In this paper, we propose a novel efficient privacy-preserving machine learning scheme for the hierarchical distributed system. Based on the analysis of hierarchical architecture, we introduce the privacy-preserving solutions to different data partition scenarios. The specific application solutions of linear support vector machine are also provided for both upper and lower layers in the system. Additionally, we present a further improvement in the learning efficiency by implementing the asynchronous learning strategy for different groups of the system. Finally, real-world datasets experiments are implemented to illustrate the privacy, efficacy, and efficiency of our approach.

Moreover, based on the analysis of the hierarchical distributed system, we can see that it has advantages than the simple distributed systems. In the normal applications of distributed learning, we can even transform the simple architecture to the hierarchical systems to improve the efficiency. For example, in a fully distributed system, we can divide the fully distributed users into different groups to abstractly build a hierarchical distributed system. Thus, the learning process will have a shrank consensus scope and the asynchronous learning process is available with different group learning speed settings. Therefore, in real applications, we can use this way to construct an abstract hierarchical system for improving the learning performance.

References

[1] D. Peteiro-Barral and B. Guijarro-Berdiñas, "A survey of methods for distributed machine learning," *Progress in Artificial Intelligence*, vol. 2, no. 1, pp. 1–11, 2013.

[2] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

[3] O. L. Mangasarian and E. W. Wild, "Privacy-preserving classification of horizontally partitioned data via random kernels." in *DMIN*, 2008, pp. 473–479.

[4] K. Chen and L. Liu, "Privacy preserving data classification with rotation perturbation," in *Fifth IEEE International Conference on Data Mining (ICDM'05)*. IEEE, 2005, pp. 4–pp.

[5] O. L. Mangasarian, "Privacy-preserving linear programming," *Optimization Letters*, vol. 5, no. 1, pp. 165–172, 2011.

[6] S. Laur, H. Lipmaa, and T. Mielikäinen, "Cryptographically private support vector machines," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 618–624.

[7] C. Orlandi, A. Piva, and M. Barni, "Oblivious neural network computing via homomorphic encryption," *EURASIP Journal on Information Security*, vol. 2007, p. 18, 2007.

[8] J. Vaidya, H. Yu, and X. Jiang, "Privacy-preserving svm classification," *Knowledge and Information Systems*, vol. 14, no. 2, pp. 161–178, 2008.

[9] H. Yu, J. Vaidya, and X. Jiang, "Privacy-preserving svm classification on vertically partitioned data," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2006, pp. 647–656.

[10] M. Santillana, A. T. Nguyen, M. Dredze, M. J. Paul, E. O. Nsoesie, and J. S. Brownstein, "Combining search, social media, and traditional data sources to improve influenza surveillance," *PLoS Comput Biol*, vol. 11, no. 10, p. e1004513, 2015.

[11] G. Tsoumakas and I. Vlahavas, "Effective stacking of distributed classifiers," in *Ecai*, vol. 2002, 2002, pp. 340–344.

[12] F. Provost and V. Kolluri, "A survey of methods for scaling up inductive algorithms," *Data mining and knowledge discovery*, vol. 3, no. 2, pp. 131–169, 1999.

[13] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 1322–1333.

[14] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *Journal of Machine Learning Research*, vol. 11, no. May, pp. 1663–1707, 2010.

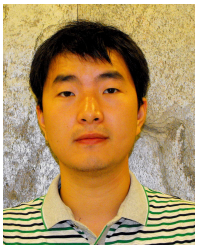
[15] K. Xu, H. Ding, L. Guo, and Y. Fang, "A secure collaborative machine learning framework based on data locality," in *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–5.

[16] K. Xu, H. Yue, L. Guo, Y. Guo, and Y. Fang, "Privacy-preserving machine learning algorithms for big data systems," in *Distributed Computing Systems (ICDCS), 2015 IEEE 35th International Conference on*. IEEE, 2015, pp. 318–327.

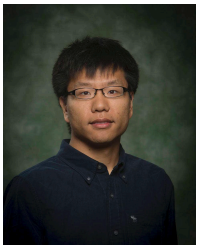
[17] K. Slavakis, G. B. Giannakis, and G. Mateos, "Modeling and optimization for big data analytics:(statistical) learning tools for our era of data deluge," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 18–31, 2014.

[18] O. Bousquet and L. Bottou, "The tradeoffs of large scale learning," in *Advances in neural information processing systems*, 2008, pp. 161–168.

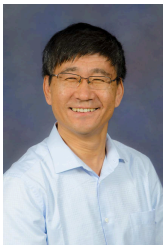
[19] M. Rodríguez, D. M. Escalante, and A. Peregrín, "Efficient distributed genetic algorithm for rule extraction," *Applied soft computing*, vol. 11, no. 1, pp. 733–743, 2011.
[20] Y. Kokkinos and K. G. Margaritis, "A distributed asynchronous and privacy preserving neural network ensemble selection approach for peer-to-peer data mining," in *Proceedings of the Fifth Balkan Conference in Informatics*. ACM, 2012, pp. 46–51.
[21] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010.
[22] J. C. Benaloh, "Secret sharing homomorphisms: Keeping shares of a secret secret," in *Conference on the Theory and Application of Cryptographic Techniques*. Springer, 1986, pp. 251–260.
[23] A. F. Karr, X. Lin, A. P. Sanil, and J. P. Reiter, "Privacy-preserving analysis of vertically partitioned data using secure matrix products," *Journal of Official Statistics*, vol. 25, no. 1, p. 125, 2009.
[24] C. Cortes and V. Vapnik, "Support vector machine," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
[25] P. Smyth, M. Welling, and A. U. Asuncion, "Asynchronous distributed learning of topic models," in *Advances in Neural Information Processing Systems*, 2009, pp. 81–88.
[26] L. Fang and Y. Lei, "An asynchronous distributed admm algorithm and efficient communication model," in *Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), 2016 IEEE 14th Intl C*. IEEE, 2016, pp. 136–140.
[27] K. Gimpel, D. Das, and N. A. Smith, "Distributed asynchronous online learning for natural language processing," in *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, 2010, pp. 213–222.
[28] T. A. Lahlou and T. A. Baran, "Asynchronous algorithms for solving linear programs," *arXiv preprint arXiv:1502.06784*, 2015.
[29] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," in *1984 American Control Conference*, 1984, pp. 484–489.
[30] Q. Ho, J. Cipar, H. Cui, S. Lee, J. K. Kim, P. B. Gibbons, G. A. Gibson, G. Ganger, and E. P. Xing, "More effective distributed ml via a stale synchronous parallel parameter server," in *Advances in neural information processing systems*, 2013, pp. 1223–1231.
[31] P. Baldi, P. Sadowski, and D. Whiteson, "Searching for exotic particles in high-energy physics with deep learning," *Nature communications*, vol. 5, 2014.
[32] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml
[33] C.-C. C. Chih-Wei Hsu and C.-J. Lin., "A practical guide to support vector classification." 2003. [Online]. Available: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/ref.html#CWH03a



Qi Jia received his B.E. degree in Communication Engineering from Beihang University (Beijing University of Aeronautics and Astronautics) in 2014. He received the MS degree electrical and computer engineering from Binghamton University and is continuing his work towards the PhD. degree. His research interests include security and privacy issues in machine learning and data mining. He is a co-recipient of Best Paper Award of Globecom 2015, Symposium on Communication and Information System Security. He is a student member of the IEEE.



Linke Guo received the BE degree in electronic information science and technology from the Beijing University of Posts and Telecommunications in 2008. He received the MS and PhD degrees in electrical and computer engineering from the University of Florida in 2011 and 2014, respectively. Since August 2014, he has been an assistant professor in the Department of Electrical and Computer Engineering, Binghamton University, State University of New York. His research interests include network security and privacy, social networks, and applied cryptography. He serves as the publication chair of IEEE Conference on Communications and Network Security (CNS) 2016 and 2017. He was the symposium co-chair of Network Algorithms and Performance Evaluation Symposium, ICNC 2016. He has served as the Technical Program Committee (TPC) members for several conferences including IEEE INFOCOM, ICC, GLOBECOM, and WCNC. He is the co-recipient of Best Paper Award of Globecom 2015, Symposium on Communication and Information System Security. He is a member of the IEEE and ACM.



Yuguang Fang (F'08) received an MS degree from Qufu Normal University, Shandong, China in 1987, a PhD degree from Case Western Reserve University in 1994, and a PhD degree from Boston University in 1997. He joined the Department of Electrical and Computer Engineering at University of Florida in 2000 and has been a full professor since 2005. He held a University of Florida Research Foundation (UFRF) Professorship (2017-2020, 2006-2009), University of Florida Term Professorship (2017-2019), a Changjiang Scholar Chair Professorship (Xidian University, Xian, China, 2008-2011; Dalian Maritime University, Dalian, China, 2015-2018), Overseas Academic Master (Dalian University of Technology, Dalian, China, 2016-2018), and a Guest Chair Professorship with Tsinghua University, China (2009-2012). Dr. Fang received the US National Science Foundation Career Award in 2001, the Office of Naval Research Young Investigator Award in 2002, the 2015 IEEE Communications Society CISTC Technical Recognition Award, the 2014 IEEE Communications Society WTC Recognition Award, and the Best Paper Award from IEEE ICNP (2006). He has also received a 2010-2011 UF Doctoral Dissertation Advisor/Mentoring Award, a 2011 Florida Blue Key/UF Homecoming Distinguished Faculty Award, and the 2009 UF College of Engineering Faculty Mentoring Award. He was the Editor-in-Chief of IEEE Transactions on Vehicular Technology (2013-present), the Editor-in-Chief of IEEE Wireless Communications (2009-2012), and serves/served on several editorial boards of journals including IEEE Transactions on Mobile Computing (2003-2008, 2011-2016), IEEE Transactions on Communications (2000-2011), and IEEE Transactions on Wireless Communications (2002-2009). He has been actively participating in conference organizations such as serving as the Technical Program Co-Chair for IEEE INFOCOM2014 and the Technical Program Vice-Chair for IEEE INFOCOM'2005. He is a fellow of the IEEE and a fellow of the American Association for the Advancement of Science (AAAS).

Guirong Wang , PhD (Dr. rer. Nat.) is a Professor in the Department of Surgery of the SUNY Upstate Medical University, Syracuse, USA. His major research interests are Cell and Molecular Biology of Pulmonary Diseases, especially for surfactant protein gene expression, regulation and biological functions, and the gene-environmental interactions using in vitro cell and in vivo transgenic mouse models. He has published 88 peer-reviewed scientific papers. His research works have been supporting by several agencies including NIH and NSF awards.