2015 IEEE 35th International Conference on Distributed Computing Systems

Privacy-preserving Machine Learning Algorithms for Big Data Systems

Kaihe Xu*, Hao Yue*, Linke Guo†, Yuanxiong Guo‡, Yuguang Fang*
*Department of Electrical and Computer Engineering,
University of Florida, Gainesville, FL 32611, USA
{xukaihe, hyue}@ufl.edu, fang@ece.ufl.edu
†Department of Electrical and Computer Engineering,
Binghamton University, Binghamton, NY 13902, USA
lguo@binghamton.edu
‡School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74078, USA
richard.guo@okstate.edu

***Abstract*—Machine learning has played an increasing important role in big data systems due to its capability of efficiently discovering valuable knowledge and hidden information. Often times big data such as healthcare systems or financial systems may involve with multiple organizations who may have different privacy policy, and may not explicitly share their data publicly while joint data processing may be a must. Thus, how to share big data among distributed data processing entities while mitigating privacy concerns becomes a challenging problem. Traditional methods rely on cryptographic tools and/or randomization to preserve privacy. Unfortunately, this alone may be inadequate for the emerging big data systems because they are mainly designed for traditional small-scale data sets. In this paper, we propose a novel framework to achieve privacy-preserving machine learning where the training data are distributed and each shared data portion is of large volume. Specifically, we utilize the data locality property of Apache Hadoop architecture and only a limited number of cryptographic operations at the Reduce() procedures to achieve privacy-preservation. We show that the proposed scheme is secure in the semi-honest model and use extensive simulations to demonstrate its scalability and correctness.**

I. Introduction

Machine learning has been widely used for scientific research and business purposes recently to extract useful information. Recent decades have witnessed accelerating development of new machine learning methods such as clustering, classification, association rule mining and sequence detection, which have been constantly improved to discover useful knowledge. However, all existing algorithms are originally designed for centralized algorithms dealing with small-scale data sets. In today's big data scenarios, things could be very different. A common scenario nowadays is that a group of organizations intend to conduct data mining over their joint data. It could be several banks wishing to conduct credit risk analysis to identify non-profitable customers based on past transaction records, or several medical institutions trying to discover certain correlations between symptoms and diagnoses from patients' records. The former case is known as data mining over vertically partitioned data [26], where the entire table is partitioned by columns and each learner has the same number of records, but the records are with different features. The latter is known as data mining over horizontally partitioned data [17], where the entire table is partitioned by rows and each learner has a certain number of rows with the same number of features. In either case, the handled data is sensitive and usually very large in its volume. In what follows, we will use data mining as the typical machine learning problems to articulate our proposed algorithms whenever needed.

This work was partially supported by US National Science Foundation under grants CNS-1423165 and CNS-1343356.

The problem of privacy-preserving data mining has been studied for decades. Numerous privacy-preserving data mining protocols have been proposed. Generally speaking, privacy-preserving data mining protocols could be classified into two categories:

- **Perturbation and randomization-based approaches** sanitize the samples prior to their release in order to mitigate the threat of inadvertent disclosure or theft [13][7]. Approaches in this category may only provide a limited privacy preservation and usually make a trade-off between utility and privacy.
- **Secure multiparty computation (SMC)-based approaches** employ cryptographic tools and focus on protocol development to protect privacy among the involved learners [9][19]. Approaches in this category may provide certain level of provable security under the standard semi-honest model, but may bring in a huge extra computation overhead[11]. This is especially true in the data mining missions that handle enormous amount of data.

Existing approaches are designed mostly for a centralized solver dealing with small-scale data sets and they may be inadequate for the emerging big data scenarios. Consider the case where a group of organizations are trying to do data

1063-6927/15 $31.00 © 2015 European Union
DOI 10.1109/ICDCS.2015.40

IEEE computer society

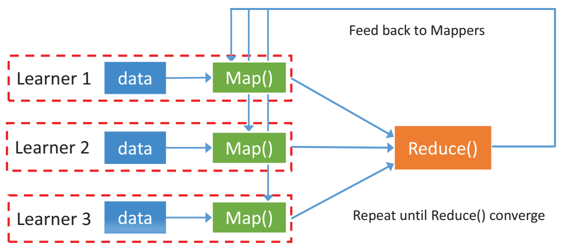


Fig. 1: System structure

mining over their joint data and each share of the data is very large in its volume. A desired scheme for this task should have two properties. First, it should be able to fit into the existing Data Parallel Systems (e.g. Apache Hadoop and/or Twister) to handle the big data. Second, cryptographic operations are minimized to guarantee training efficiency.

Data Parallel Systems are crucial to the success of big data. In Data Parallel Systems such as Google File Sysatem(GFS) and MapReduce, clusters are built with commodity hardware and each node takes the roles of both computation and storage, which is known as data locality. Data locality is a significant advantage of data parallel systems over traditional High Performance Computing(HPC) systems because moving computation result is much cheaper than moving data. We observe that another potential big advantage of data locality is its ability to protect the local private data. Imagine that if machine learning could be done with distributed local learners independently and somehow a consensus learning result could be deduced from the local learning results, there is no privacy concerns about the local training data because they are controlled by the local learners and never leave the local domain. For example, in MapReduce framework, local training could be regarded as the Map() procedures and the consensus forming process be the Reduce() procedures. With data locality, security bottleneck of the whole system lies in Reduce() because Map() operates independently with local data. Notice that Reduce() is usually much simpler than Map(), and then a few simple and efficient cryptographic operations are sufficient. In the rest of this paper we use Map() and Mapper interchangeably to denote the local training procedure; Reduce() and Reducer interchangeably to denote the consensus procedure.

In this paper, we propose a framework based on MapReduce and data locality is also used to conceal the local training data. Our system structure is illustrated in Fig. 1. Each learner is treated as a data node of HDFS (Hadoop Distributed File System). Each data node has a Mapper who will perform machine learning over the local data to get a local training result. The local training result is then sent to Reducer. A secure protocol is implemented on the Reducer such that the local training results could be summarized securely. Notice that it may require a few back and forth negotiation processes for the local Mappers to reach a consensus. Hence, there should exist a feedback channel from the Reducer to Mappers to feed back the current negotiated results. We focus on support vector machine and propose two schemes for horizontally and vertically training sets.

The rest of this paper is organized as follows: In Section II we discuss the existing approaches and compare them with our approach. In Section III, we briefly discuss support vector machines (SVMs), kernel-based nonlinear SVM, and alternating direction method of multipliers (ADMM). In Section IV, we present the designed schemes for horizontally partitioned case and vertically partitioned case, respectively. In Section V, security analysis is conducted. In Section VI, the performance of our scheme is tested against three popular data sets and finally, in Section VII, we conclude this paper.

II. Related Works

Privacy-preserving data mining falls into two major categories: randomization-based approaches and SMC-based approaches. Randomization-based approaches protect the private data by adding noise to the original data. In [21] and [22], both the horizontally and vertically partitioned data scenarios are studied for the SVMs. In their schemes, a randomly generated matrix known as "random kernel" is directly multiplied to the original kernel matrix. It is argued that random kernels will hide the private data while guaranteeing accurate learning results. This method works because some properties are preserved by matrix multiplication which is also known as the restricted isometry property (RIP) in the compressive sensing research. The drawbacks of the random kernel approaches is that the random kernels should be shared among the learners as a common key and it only works under the client and server scenario. In [7], Chaudhuri and Monteleoni show that one can either add noise to the final data mining results or directly work on a perturbed objective function to guarantee that the original training data is ϵ-differentially private. The ϵ-differential privacy model limits the amount of information an adversary can gain about a particular private value by observing a function learned from a database containing that value, even if she knows every other value in the database. However, this model has a restricted requirement: the objective function and loss function must be differentiable everywhere with continuous derivatives and convex. In [1], Agrawal and Srikant show that adding random noise to the training data still preserve some statistical properties. As a result, a Naive Bayes classifier could still be obtained from the sanitized data. In [14], Fong and Weber-Jahnke propose a method to generate some fake training data set from the real ones, while useful knowledge could still be discovered from the fake training sets. It has been claimed that the privacy and utility are achieved at the same time. However, the generation of the fake training sets induces huge extra cost.

The SMC-based approaches usually rely on some specific design of a certain learning method. For example, in [30], Yuan and Yu use homomorphic encryption to calculate the delta function in the back-propagation training; in [28] and [31], Jiang and Zhan propose to use secure dot product protocols and secure sum protocols to get the kernel matrix in SVM learning; in [20], Lindell and Pinkas propose a secure protocol to compute the result of $(v_1+v_2)\log(v_1+v_2)$ without revealing v_1 and v_2 to each other, which is the key step for the ID3 algorithm in building a decision tree; in [18] Kantarcioglu and Clifton use commutative encryption and secure sum protocols to find the association rules over the horizontally partitioned data and in [27], Vaidya and Clifton apply secure dot protocols to find association rules over the vertically partitioned data. Generally speaking, SMC based approachs use secure protocols on a few key steps of a particular machine learning algorithm and send the SMC results to a centralized node to perform learning.

III. Preliminaries

A. Linear Support Vector Machines(SVMs)

SVMs were first introduced in [10]. Basically, SVMs are working towards an optimal separating hyper-plane between two classes of data. The two classes are determined as optimally separated when the margin between them is maximized, which is defined as the distance from the closest point of one class to the boundary of the other class. The optimization problem for SVM could be formulated as

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}w^T w + C\|\xi\|_1^1 \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, \forall i \end{aligned} \tag{1}$$

where each x_i is one training sample of size $k \times 1$ and y_i is the corresponding class label. ξ is an $N \times 1$ slack variable vector whose entries are all non-negative. The slack variable ξ could be used to reject outliers. When the two classes are not linearly separable, we can use the parameter C to make a tradeoff between maximum margin and the tolerance of outliers. Usually, a support vector machine problem (1) is approached through KKT conditions and its Wolfe-dual, which is in the following form:

$$\begin{aligned} \min_{\lambda} \quad & \frac{1}{2}\lambda^T H \lambda - \mathbf{1}^T \lambda \\ \text{s.t.} \quad & \mathbf{0} \leq \lambda \leq C\mathbf{1}, \\ & \lambda^T y = 0. \end{aligned} \tag{2}$$

In problem (2), **0** is a vector of all zeros and **1** is a vector of all ones and y denotes the vector of labels. The matrix H is constructed as $H_{ij} = y_i x_i^T x_j y_j$. The optimal values of λ indicate the support vector. If $0 < \lambda_i \leq C$, x_i is found as a support vector. Instead of solving problem (2) directly through a quadratic programming solver, there exist a number of faster algorithms such as Osuna[23] and SMO(sequential minimal optimization)[24] used in LIBSVM[6].

After problem (2) is solved, w is calculated as $w = \sum_{i \in SV} \lambda_i x_i y_i$, where SV denotes the set of support vectors. With support vectors, b could be found from $y_i(w_i x_i + b) = 1, \forall i \in SV$. Notice that we may obtain multiple values for b due to multiple support vectors. In [5], Burges suggest to use the average value of b resulting from all the support vectors. However, in some SVM implementation such as Matlab and LIBSVM[6], they only use one support vector to find b which is the one corresponding to the maximum λ. To the best of our knowledge, these two methods have similar performance.

B. Nonlinear SVM with Kernel Tricks

The general task for machine learning is to find and study the relationship between data points. Kernel methods provide a valuable way to access the similarity in a high-dimensional implicit feature space without ever computing the coordinates in that feature space, which will greatly reduce the computation. Specifically, for $x_i, x_j \in \mathbb{R}^k$, $\phi(\cdot) : \mathbb{R}^k \to \mathbb{R}^p$, denote the mapping of x from low-dimensional $\mathbb{R}^k$ to a high-dimensional Reproducing Kernel Hilbert Space (RKHS) $\mathbb{R}^p$. The explicit form of a kernel function $K : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}$ is defined as:

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \tag{3}$$

Three most popular kernels, i.e., polynomial, radial basis function, and sigmoid kernels, are listed as follows:

1) Polynomial Kernel: $K(x_i, x_j) = (a\langle x_i, x_j \rangle + b)^d$
2) Radial Basis Function Kernel: $K(x_i, x_j) = e^{\|x_i - x_j\|^2}$
3) Sigmoid Kernel: $K(x_i, x_j) = \tanh(\langle x_i, x_j \rangle + c)$

Nonlinear SVM utilizes kernel tricks and writes its discriminant function as $f(x) = w^T \phi(x) + b_m$. However, w is in the unknown high-dimensional space and is hard to find. For example, RBF kernel will lead to an infinite-dimensional w. In that sense, the primary problem of SVM is not available. However, if we follow the standard Wolfe-dual transformation to find the dual problem, we could play with kernel tricks to get rid of the high-dimensional vectors. As shown in [5], nonlinear SVMs could still be solved from problem (2). The only difference is that H is obtained by $H_{ij} = y_i K(x_i, x_j) y_j$. Then, with the dual variables λ, the discriminant function is written as $f(x) = \sum_{i \in SV} \lambda_i y_i K(x_i, x)$.

C. ADMM

The alternating direction method of multipliers (ADMM) is a variant of the augmented Lagrangian scheme that uses partial updates for the dual variables (dual ascent). It is intended to blend the decomposability of dual ascent with the superior convergence properties of the augmented Lagrangian scheme[4]. Generally, ADMM is designed for a problem in the following form:

$$\begin{aligned} \min_{x,z} \quad & f(x) + g(z) \\ \text{s.t.} \quad & Ax + Bz = c. \end{aligned} \tag{4}$$

ADMM approaches to the optimal solution of x and z through the following iterations:

$$\begin{aligned} x^{t+1} &:= \text{argmin} \quad f(x) + \frac{\rho}{2}\|Ax + Bz^t - c + u^t\|_2^2 \\ z^{t+1} &:= \text{argmin} \quad g(z) + \frac{\rho}{2}\|Ax^{t+1} + Bz - c + u^t\|_2^2 \\ u^{t+1} &:= u^t + Ax^{t+1} + Bz^{t+1} - c \end{aligned} \tag{5}$$

In this problem, the variable u could be understood as the dual variable, according to the dual ascent algorithm. On each iteration, only partial updates are applied to u until convergence, i.e., $u = 0$. It is this partial update of the dual variable that enables us to calculate x and z independently, even though they are correlated with each other by an equality constraint in problem (4). With this nice property, we may be able to

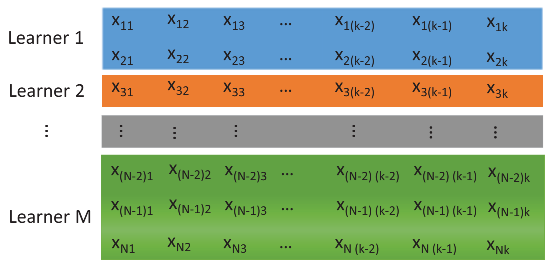

Learner 1
Learner 2
Learner M
x_{11} x_{12} x_{13} … $x_{1(k-2)}$ $x_{1(k-1)}$ x_{1k}
x_{21} x_{22} x_{23} … $x_{2(k-2)}$ $x_{2(k-1)}$ x_{2k}
x_{31} x_{32} x_{33} … $x_{3(k-2)}$ $x_{3(k-1)}$ x_{3k}
$x_{(N-2)1}$ $x_{(N-2)2}$ $x_{(N-2)3}$ … $x_{(N-2)(k-2)}$ $x_{(N-2)(k-1)}$ $x_{(N-2)k}$
$x_{(N-1)1}$ $x_{(N-1)2}$ $x_{(N-1)3}$ … $x_{(N-1)(k-2)}$ $x_{(N-1)(k-1)}$ $x_{(N-1)k}$
x_{N1} x_{N2} x_{N3} … $x_{N(k-2)}$ $x_{N(k-1)}$ x_{Nk}

Fig. 2: The training set is horizontally partitioned

decompose problem (4) into two subproblems and distribute the computation task. Meanwhile, the term $\frac{\rho}{2}\|Ax+Bz^t-c+u^t\|_2^2$ guarantees the convergence property.

IV. The Proposed Approach

In this section, we first present our scheme with linear SVMs of horizontally partitioned training data. Then, a more sophisticated version is proposed for nonlinear SVMs. More over, we will also show how to modify our scheme to work under a vertically partitioned scenario. In order to take advantage of data locality to protect private training data, the proposed scheme should be managed to 1) decompose the joint learning task into independent small subtasks; 2) assign subtasks to Mappers such that Mappers could perform local training without knowing others' private data.

A. Linear SVM with Horizontally partitioned data

A horizontally partitioned scenario could be visualized as Fig. 2. The joint training set X includes totally N records shown as N rows, and each record is a training sample with k attributes. N records are distributedly located at to M learners. Let X_m denote the training set at learner m of size $N_m \times k$. Y_m is a square matrix whose diagonal elements are the corresponding labels. w_m and b_m are local training results. Consider the following problem:

$$\begin{aligned} \min_{\{w_m\},\{b_m\}} \quad & \sum_m^M \frac{1}{2M} w_m^T w_m + C\sum_m^M \|\xi_m\|_1^1 \\ \text{s.t.} \quad & Y_m(X_m w_m + \mathbf{1} b_m) \geq \mathbf{1} - \xi_m, \\ & \xi_m \geq \mathbf{0}, \\ & w_m = z, \\ & b_m = s \quad \forall m = 1,2,...M. \end{aligned} \tag{6}$$

Lemma 4.1: The optimal solution of problem (1) is identical to the solution of problem (6), i.e., $w = w_m, b = b_m, \forall m = 1, 2...M$.

Proof: Assume the constraints of $w_m = z$ and $b_m = s, \forall m = 1,2,...M$ are all strictly enforced. Then, it is safe to substitute $\{w_m\}$ with z, $\{b_m\}$ with s, and omit the last two equality constraints. Then, we will obtain an optimization problem with the same objective function and constraints as problem (1). Hence, the solutions of the two problems are identical. ■

The next step is to decompose problem (6) into subproblems that fit MapReduce framework. Firstly, we can see that the objective function is decomposable because if we take partial derivative of the objective function, $w_n, \forall n \neq m$ will vanish. Secondly, the constraints could be divided into groups such that each group only related to a single w_m. With these two observations, the subtasks are independent if the subtask assigned to Mapper m is defined as:

$$\begin{aligned} \min_{w_m,b_m} \quad & \frac{1}{2M} w_m^T w_m + C\|\xi_m\|_1^1 \\ \text{s.t.} \quad & Y_m(X_m w_m + \mathbf{1} b_m) \geq \mathbf{1} - \xi_m, \\ & \xi_m \geq \mathbf{0}, \\ & w_m = z, \\ & b_m = s. \end{aligned} \tag{7}$$

However, problem (7) is still hard to solve. The equality constraints of $w_m = z$ and $b_m = s$ are difficult to enforce for all the M learners at the same time because we do not know the values of z and s. Actually, z and s are the optimal solution to the centralized problem (1). Instead of requiring each subtask to enforce the equality constraints immediately, we could introduce a regularization term of $\sum_m^M \frac{\rho}{2}\|w_m - z\|_2^2$ to the objective function. By doing this, $\{w_m\}$ are optimizing the inverse margin while approaching to z. After this, z could be updated with the calculated $\{w_m\}$. This cyclic optimization is iterating until convergence. Similar idea will be applied to $\{b_m\}$. As a summary, with the regularization terms, Lagrangian function of problem (6) could be written in the following form as the augmented Lagrangian:

$$\begin{aligned} \mathcal{L} = \sum_m^M \{ & \frac{1}{2M} w_m^T w_m - \lambda_m[Y_m(X_m w_m + \mathbf{1} b_m) - \mathbf{1} + \xi_m] \\ & - \mu_m \xi_m + C\|\xi_m\|_1^1 + \frac{\rho}{2}\|w_m - z + \gamma_m\|^2 \\ & + \frac{\rho}{2}\|b_m - s + \beta_m\|^2 \} \end{aligned} \tag{8}$$

In the augmented Lagrangian (8), γ_m and β_m are introduced to get rid of the dual variables for $w_m = z$ and $b_m = s$ [4]. With function (8), alternative optimizations of w_m, z, b_m, and s could be carried out as:

$$\{w_m^{t+1}\} = \mathbf{arg} \min_{\{w_m\}} \mathcal{L}(\{w_m\}, z^t, \{b_m^t\}, s^t) \tag{9a}$$

$$z^{t+1} = \mathbf{arg} \min_z \mathcal{L}(\{w_m^{t+1}\}, z, \{b_m^t\}, s^t) \tag{9b}$$

$$\{b_m^{t+1}\} = \mathbf{arg} \min_{\{b_m\}} \mathcal{L}(\{w_m^{t+1}\}, z^{t+1}, \{b_m\}, s^{t+1}) \tag{9c}$$

$$s^{t+1} = \mathbf{arg} \min_s \mathcal{L}(\{w_m^{t+1}\}, z^{t+1}, \{b_m^{t+1}\}, s) \tag{9d}$$

Notice that in problem (9a), w_m is obtained through its dual problem by first solving λ_m as follows:

$$\begin{aligned} \mathcal{L}_d = & -\frac{M}{2(1+\rho M)} \lambda_m Y_m X_m X_m^T Y_m \lambda_m^T \\ & + [\mathbf{1} - \frac{M\rho}{1+\rho M}(z^t - \gamma_m^t) Y_m X_m]^T \lambda_m \end{aligned} \tag{10}$$

Let

$$\begin{aligned} A &= \frac{M}{1+\rho M} Y_m X_m X_m^T Y_m \\ B^t &= -\mathbf{1} + \frac{M\rho}{1+\rho M}(z^t - \gamma_m^t) Y_m X_m \end{aligned} \tag{11}$$

Then, the duel problem could be written in the following nice and neat form:

$$\begin{aligned} \lambda_m :=& \mathbf{arg} \min_{\mathbf{0} \le \lambda_m \le \mathbf{C}} \frac{1}{2} \lambda_m A \lambda_m^T + B^T \lambda_m \\ &\text{s.t.} \quad \mathbf{1} Y_m \lambda_m = \rho(b_m^t - s^t + \beta_m^t) \end{aligned} \tag{12}$$

Problem (12) could be solved by a standard quadratic programming solver. We can also adapt the existing improved SVM solver to solve this problem. For example, SMO (Sequential Minimal Optimization) is still feasible, because the relationship between any two entries of λ_m is defined with the constraint $\lambda_m Y_m = \rho(b_m^t - s^t + \beta_m^t)$. The adaption of SMO to our solution is orthogonal to this paper, which can be easily taken into our algorithm. We will use Wolfe-dual and a quadratic programming solver to find λ_m. With λ_m, the iterative variable update could be summarized as follows:

$$w_m^{t+1} = \frac{M}{1+\rho M}(\rho z^t - \rho \gamma_m^t + \lambda_m^{t+1} Y_m X_m) \tag{13a}$$

$$z^{t+1} = \frac{1}{M}\left(\sum_n^M w_m^{t+1} + \sum_m^M \gamma_m^t\right) \tag{13b}$$

$$\gamma_m^{t+1} = \gamma_m^t + w_m^{t+1} - z^{t+1} \tag{13c}$$

$$b_m^{t+1} = \frac{1}{\rho} \lambda_m^{t+1} Y_m + s^t - \beta_m^t \tag{13d}$$

$$s^{t+1} = \frac{1}{M}\left(\sum_n^M b_m^{t+1} + \sum_m^M \beta_m^t\right) \tag{13e}$$

$$\beta_m^{t+1} = \beta_m^t + b_m^{t+1} - s^{t+1} \tag{13f}$$

Notice that in (13c) and (13f), gradient descent is used to update γ_m and β_m. Lemma 4.2 shows that if we update the parameters according to (13), $\{w_m\}$ and $\{b_m\}$ will converge to the optimal solution.

Lemma 4.2: By following iterative update (13), $\{w_m\}$ and $\{b_m\}$ will converge to the optimal solution.

Proof: Problem (6) could be written in a general form as:

$$\begin{aligned} \min_{w_1, w_2} \quad & F_1(w_1) + F_2(w_2) \\ \text{s.t.} \quad & A w_1 = w_2, \\ & w_1 \in \mathcal{S}_1, \ w_2 \in \mathcal{S}_2. \end{aligned} \tag{14}$$

Here, $\mathcal{S}_1 = \{w | Y_1(X_1 w_1 + b_1) \ge 1 - \xi_1\}$ and $\mathcal{S}_2 = \{w | Y_2(X_2 w_2 + b_2) \ge 2 - \xi_2\}$ are the feasible sets of w_1 and w_2. It has been proved in [15] and [4] that the convergence of problem (14) is guaranteed as long as one of the following two conditions is true: $\mathcal{S}_1$ and $\mathcal{S}_1$ are bounded; or AA^T is nonsingular. In our problem, A is an identity matrix, thus w_1 and w_2 will converge. ■

Take a closer look at the iterative update procedures (13), w_m^t is only related to the local training set of X_m, Y_m and the global consensus variable z^{t-1}. Hence, each w_m^t could be updated in parallel by the local Mappers. A global Reducer is in charge of updating z^t by collecting the local training results and finding the average of them. Actually, the Reducer does not need to know the exact value of each local training result as long as it is able to find the average of these local training results. In section V, we present a secure summation protocol at the Reducer to find the average value of local learning results securely. After the Reducer finds z^t, it sends z^t back to each Mapper to update w_m^{t+1}. This iterative update will continue until z converges. Notice that Hadoop MapReduce may not be suitable for iterative computation tasks, however, in [12], Ekanayake and Li propose an iterative version of MapReduce called Twister, which could be used to achieve iterative update in (13).

B. Nonlinear SVM with Horizontally Partitioned Data

Non-linear version of the proposed scheme is not a trivial modification of its linear version. Each Mapper may be able to find the solution of the involved nonlinear SVM problem by playing with kernel tricks as shown in Subsection III-B. However, it could be very hard for Reducer to summarize the local learning results. The reason is that while solving the nonlinear SVM problem, the local learning result w_m has never been calculated explicitly. For example, with RBF kernels, w_m is an infinite dimensional vector. Kernel tricks are not trying to find infinite dimensional vector, but they directly find the discriminant function as $f(x) = \sum_{x_i \in SV} K(x_i, x) + b$. Hence, a desired feature of our nonlinear version should be the ability to summarize local learning results without knowing them explicitly.

Kernel functions map the training data to a high-dimensional space $\mathbb{R}^p$ whose complexity prohibits us from exchanging the local learning result. However, although consensus in $\mathbb{R}^p$ is hard to achieve, we may modify the consensus constraint to achieve consensus in a reduced space $\mathbb{R}^l$, where $l < p$ as $G w_m^T = z$. In this sense, G is an $l \times p$ matrix and z is the global consensus of size $1 \times l$. Then, problem (6) is rewritten as:

$$\begin{aligned} \min_{\{w_m\}} \quad & \sum_m^M \frac{1}{2M} \|w_m\|^2 + C \sum_m^M \|\xi_m\|_1^1 \\ \text{s.t.} \quad & Y_m[\phi(X_m) w_m - \mathbf{1} b_m] \ge \mathbf{1} - \xi_m, \\ & \xi_m \ge \mathbf{0}, \\ & G w_m = z, \\ & b_m = b \qquad \forall m = 1, 2, ...M, \end{aligned} \tag{15}$$

where $\phi(\cdot) : \mathbb{R}^k \to \mathbb{R}^p$ is the mapping from a low-dimensional space to a high-dimensional space. Assume $\phi(X_m)$ is an $N_m \times p$ matrix as the representation of each row of X_m in $\mathbb{R}^p$. The dimension-reduction matrix G is generated by $G = \phi(X_g)$, where X_g is an $l \times k$ matrix of rank $\min(l, k)$. Each row of G is generated by mapping one row of X_g to $\mathbb{R}^p$.

Problem (15) is still hard to solve, because each row of G is in $\mathbb{R}^p$ which we cannot directly operate on. However, recall that the final goal for SVM is not to find w_m and b_m, but rather to find the discriminant function $f_m(x) = w_m^T \phi(x) + b_m$. The following lemmas show that problem (15) could be approximately solved and suggest a general form for its discriminant function.

Lemma 4.3: For a fixed kernel $K(\cdot,\cdot)$ and the corresponding RKHS $\mathbb{R}^p$, if the SVM optimization problem could be expressed with a loss function $L:\mathbb{R}^n\to\mathbb{R}$ and a non-decreasing function $\Omega:\mathbb{R}\to\mathbb{R}$ in the following form,

$$\min_{f\in\mathbb{R}^p} L(f(x_1),f(x_2),...f(x_n))+\Omega(\|f\|^2), \tag{16}$$

the functional $f(\cdot)$ could be expressed as $f(\cdot)=\sum_i^n \alpha_i K(x_i,\cdot)$.

Proof: See the representer theorem[25]. ■

Lemma 4.4: Applying Lemma.4.3 to problem (15), we have

$$f_m(\cdot)=\sum_{x_i\in X_m} a_i K(x_i,\cdot)+\sum_{x_j\in X_g} c_j K(x_j,\cdot). \tag{17}$$

Proof: For problem (15), we can find a loss function as $L:\mathbb{R}^{N_m+l}\to\mathbb{R}$. Among its N_m+l entries, the first N_m entries represent the loss for N_m inequality constraints and the last l entries represent the loss for the l equality constraints of $Gw=z$. The max-margin objective function could be written as a non-decreasing function as $\frac{1}{2M}\|f_m\|^2$. ■

Lemma. 4.4 also shows how problem (15) approximately summarizes the local learning results: since $\{w_m\}$ lie in $\mathbb{R}^p$, the true consensus result $\tilde{w}$ should also lie in $\mathbb{R}^p$. If we have p independent vectors in $\mathbb{R}^p$, we can perfectly reconstruct $\tilde{w}$. However, because we cannot afford p vectors, we only use l vectors to approximate $\tilde{w}$ as $\sum_{x_j\in X_g} c_j K(x_j,\cdot)$. In section VI, experiments on real-life data show that this approximation gives reasonably good performance. Next, we will show how to follow the similar process as in the previous subsection to find the detailed iterative variable update. Firstly, find the augmented Lagrangian as:

$$\begin{aligned}\mathcal{L}=\sum_m^M\{&\frac{1}{2M}\|w_m\|^2-\mu_m^T\xi_m+\frac{\rho}{2}\|Gw_m-z+r_m\|^2\\&+C\|\xi_m\|-\lambda_m^T[Y_m(\phi(X_m)w_m-\mathbf{1}b_m)-\mathbf{1}+\xi_m]\\&+\frac{\rho}{2}\|b_m-b+s_m\|^2\}\end{aligned} \tag{18}$$

w_m could be found by taking partial derivative of (18) and setting it to zero:

$$\begin{aligned}&\frac{\partial\mathcal{L}}{\partial w_m}=0\\&\Rightarrow w_m=(I+\frac{\rho}{M}G^TG)^{-1}[\phi(X_m)^TY_m\lambda_m\\&\qquad+\rho G^T(z-r_m)].\end{aligned} \tag{19}$$

Notice that the term G^TG is a $p\times p$ matrix and we do not know anything in $\mathbb{R}^p$. However, with Sherman-Morrison-Woodbury formula [16], $(I+\rho/MG^TG)^{-1}$ could be rewritten as

$$(I+\frac{\rho}{M}G^TG)^{-1}=I-\frac{\rho}{M}G^T(I+\frac{\rho}{M}GG^T)^{-1}G. \tag{20}$$

Let K_g denote $I+\frac{\rho}{M}GG^T$, by substituting (20) into (19), we can find w_m as:

$$\begin{aligned}w_m=&\phi(X_m)^TY_m\lambda_m-\frac{\rho}{M}G^TK_g^{-1}K(X_g,X_m)Y_m\lambda_m\\&+\rho G^T(z-r_m)-\frac{\rho^2}{M}G^TK_g^{-1}K(X_g,X_g)(z-r_m)\end{aligned} \tag{21}$$

Notice that the above equation holds because $G\phi(X_m)^T=K(X_g,X_m)$ and $GG^T=K(X_g,X_g)$. By substituting w_m back to (18), the Wolfe-dual could be written as:

$$\begin{aligned}\lambda_m:=&\mathbf{arg}\min_{\mathbf{0}\le\lambda_m\le\mathbf{C}}\ \frac{1}{2}\lambda_mA\lambda_m^T+B^T\lambda_m\\&\text{s.t. }\ \mathbf{1}Y_m\lambda_m=\rho(b_m^t-s^t+\beta_m^t)\end{aligned} \tag{22}$$

where

$$\begin{aligned}&A=Y_m[K(X_m,X_m)-\frac{\rho}{M}K(X_m,X_g)K_g^{-1}K(X_g,X_m)]Y_m\\&B=I-\rho Y_mK(X_m,X_g)(z-r_m)\\&\quad+Y_m\frac{\rho^2}{M}K(X_m,X_g)K_g^{-1}K(X_g,X_g)(z-r_m)\end{aligned}$$

Then the local training result mapped by G in $\mathbb{R}^k$ is calculated by

$$\begin{aligned}z_m^{t+1}&=Gw_m^{t+1}+r_m^t\\&=K(X_g,X_m)Y_m\lambda_m^{t+1}+\rho K(X_g,X_g)(z^t-r_m^t)\\&-\frac{\rho}{M}K(X_g,X_g)K_g^{-1}K(X_g,X_m)Y_M\lambda_m\\&-\frac{\rho^2}{M}K(X_g,X_g)K_g^{-1}K(X_g,X_g)(z^t-r_m^t)+r_m^t\end{aligned} \tag{23}$$

Finally, the local training results are sent to the Reducer, which will use a secure summation protocol to find the global consensus as $z^{t+1}=\frac{1}{M}\sum_m^M z_m^{t+1}$. The iterative update is summarized as follows:

$$\lambda_m^{t+1}:\ \text{Found by solving (22)} \tag{24a}$$
$$z_m^{t+1}:\ \text{Calculated by (23)} \tag{24b}$$
$$\gamma_m^{t+1}=\gamma_m^t+z_m^{t+1}-z^{t+1} \tag{24c}$$
$$z^{t+1}=\frac{1}{M}\sum_m^M z_m^{t+1} \tag{24d}$$
$$b_m^{t+1}=\frac{1}{\rho}\lambda_m^{t+1}Y_m+s^t-\beta_m^t \tag{24e}$$
$$s^{t+1}=\frac{1}{M}(\sum_n^M b_m^{t+1}+\sum_m^M\beta_m^t) \tag{24f}$$
$$\beta_m^{t+1}=\beta_m^t+b_m^{t+1}-s^{t+1} \tag{24g}$$

Notice that convergence of iterative update (24) could be proved by Lemma 4.2 as long as GG^T is non-singular, i.e. X_g could be randomly chosen such that $K(X_g,X_g)$ is non-singular. After training, learner m could classify a testing sample x as:

$$\begin{aligned}f_m(x)=&K(x,X_m)Y_m\lambda_m+\rho K(x,X_g)(z-r_m)\\&-\frac{\rho}{M}K(x,X_g)K_g^{-1}K(X_g,X_m)Y_m\lambda_m\\&-\frac{\rho^2}{M}K(x,X_g)K_g^{-1}K(X_g,X_g)(z-r_m)+b_m\end{aligned} \tag{25}$$

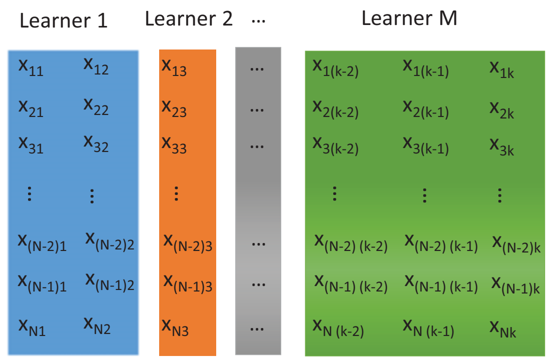


Fig. 3: The training set is vertically partitioned

C. *Vertically Partitioned Data*

In this subsection, we present our scheme under the situation where the training data is vertically partitioned. The vertically partitioned data could be visualized as shown in Fig. 3: each learner has N records, however, for each record, they hold different attributes, i.e., the k attributes are distributed among M learners.

The local learning task in this scenario is different from that in the horizontally partitioned case. This is because of the following several reasons. 1). The label $y_i, \forall i = 1, 2...N$ should be agreed and shared among M learners. 2). Each learner only has a share of w denoted as w_m of size $N_m \times 1$. 3). The constraints are no longer separable because each inequality constraint in problem (1) requires data from all the M learners. Following the similar distribute-and-consensus idea as shown in the previous section, we reformulate the problem as follows:

$$\begin{aligned} \min_{\{w_m\}} \quad & \sum_m^M \frac{1}{2}\|w_m\|^2 + C\|\xi\|_1^1 \\ \text{s.t.} \quad & Y(z+b) \geq \mathbf{1} - \xi, \\ & \xi > \mathbf{0}, \\ & z = \sum_m^M X_m w_m. \end{aligned} \tag{26}$$

In this problem, Y is an $N \times N$ diagonal matrix, whose diagonal elements are the labels. z is an $N \times 1$ vector works as the decoupling item. With z, the inequality constraint no longer depends on training data from all the learners, which makes it possible to decompose the problem. Solutions to problem (26) and problem (1) are identical because the objective functions and constraints are identical. We could use similar method to find an iterative update of problem (26). Firstly, find the augmented Lagrangian as:

$$\begin{aligned} \mathcal{L} = & \sum_m^M \frac{1}{2}\|w_m\|^2 + \frac{\rho}{2}\|z - \sum_m^M X_m w_m + \gamma\|_2^2 \\ \text{s.t.} \quad & Y(z+b) \geq \mathbf{1} - \xi, \end{aligned} \tag{27}$$

where γ is the residual term. An alternative approach is used to optimize (27) and the iterative updates are listed as follows:

$$\{w_m^{t+1}\} = \mathbf{arg} \min_{\{w_m\}} \mathcal{L}(\{w_m\}, z^t, \gamma^t) \tag{28a}$$

$$z^{t+1} = \mathbf{arg} \min_{z} \mathcal{L}(\{w_m^{t+1}\}, z, \gamma^t) \tag{28b}$$

$$\gamma^{t+1} = \gamma^t + z^{t+1} - \sum_m^M X_m w_m. \tag{28c}$$

The solution to problem (28) is straightforward: w_m has a closed-form by taking the partial derivative and setting it to zero; z could be solved via its Wolfe-dual problem. The solution is listed as follows:

$$\begin{aligned} w_m^{t+1} = & \rho(\mathbf{1} + \rho X_m^T X_m)^{-1} X_m^T [z - \bar{c} + c_m^t + r^t] \\ \lambda = & \mathbf{arg} \min_{\mathbf{0} \leq \lambda \leq C} \frac{1}{2}\lambda^T A \lambda + B^T \lambda \\ & \text{s.t. } \mathbf{1} Y \lambda = 0, \\ z^{t+1} = & \bar{c}^{t+1} - r^t + \frac{1}{\rho}\lambda Y \\ r^{t+1} = & r^t + z^{t+1} - \bar{c}^{t+1}, \end{aligned} \tag{29}$$

where we write $\bar{c}^t = \sum_m^M X_m w_m^t$ and $c_m^t = X_m w_m^t$ for the sake of simplicity, $A = \frac{1}{\rho} Y \mathbf{1}\mathbf{1}^T Y$, $\mathbf{1}$ is a $N \times 1$ vector of all ones, and $B = -\mathbf{1} + Y(\bar{c}^{t+1} - r^t)$.

In this scenario, the Reducer is in charge of calculating $\bar{c}$ with a secure summation protocol and finding z by solving problem (28b) through its dual problem. The nonlinear version under vertically partitioned scenario is a straightforward modification, because the global consensus in this scenario is z of a fixed size independent of the kernel functions used. We only need to substitute $(I + \rho X_m^T X_m)^{-1}$ with a similar form as in (20) and play with kernel tricks.

V. SECURITY ANALYSIS

There are two fundamentally different ways a private-preserving machine learning scheme can disclose sensitive information: (a) side information during the run time is collected and analyzed by the adversary; (b) the specified result itself reveals sensitive aspects of the training data. For the first issue, we need to be very careful about the information that could be possibly revealed and the consequence. For example, in [8] and [29], although the kernel is calculated in a privacy-preserving manner (secure dot product), if the kernel matrix is obtained by a learner with more than k training samples, he can calculate all the private training samples of the other learners by solving a set of linear equations, where k is the number of features. For the second issue, clearly, there always exists a tradeoff between revealing sensitive information and utility. A certain amount of information is inevitably lost during training.

In this paper, we assume that: 1). The collaborative learners work in a semi-honest manner, i.e., they will follow the protocol to obtain the joint training result, but they are curious about the private training sets of other parties. 2). The learners understand the tradeoff between privacy and utility and agree that the joint machine learning result does not reveal their private training sets. However, if the local training result of each iteration is collected, an adversary may be able to reverse engineer the private training set X_m. Hence, w_m^t is considered sensitive and should not be revealed. 3). The Reducer also works in a semi-honest manner, i.e., it is curious about the local training results w_m^t but will follow the protocol to summarize them. 4). Due to data locality, Map() is a local operation and is trusted by each learner.

With these assumptions, our scheme is secure as long as the local training results are averaged without disclosing each individual value. In this paper, we use a coalition-resistance secure summation protocol to average the local training result as:

Protocol Coalition-resistance secure summation protocol

```
1: Each Mapper generates $M$-1 random numbers.
2: For each Mapper, the $M$-1 numbers are sent to other $M$-1 Mappers individually.
3: Each Mapper $i$ sums over its generated numbers as $Sed_i$ and its received number as $Rev_i$.
4: Each Mapper $i$ sends $w_i + Sed_i - Rev_i$ to the Reducer.
5: Reducer averages the received value to find $z$.
```

With this protocol, the Reducer is able to find the average value because every generated random number is added once and subtracted once. The individual local training results are hidden by $Sed_i - Rev_i$ and coalition attack is prevented, i.e. the individual training result is secure even if a group of Mappers coalesce to attack one Mapper.

For the vertically partitioned scheme, the Reducer is in charge of solving problem (26) to find z. The information available to him is labels Y and $\bar{c}$, where Y is shared by all the learners and $\bar{c}$ is calculated by our secure summation protocol to hide $X_m w_m$. Generally speaking, our scheme is secure because: 1). Private data is only processed locally such that the data holder never loses control over its data. 2). The process of machine learning is hard to reverse, i.e., given w_m, X_m is still very hard to find. 3). To make the reverse engineering even harder, $\{w_m\}$ are mixed together with a secure summation protocol.

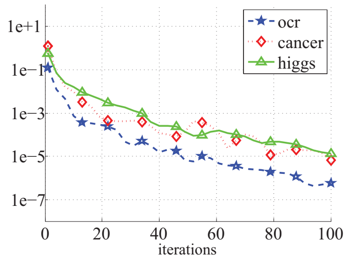
VI. Performance Analysis

In this section, we use three popular data sets to test the performance of our scheme in terms of the convergence and correctness. The three data sets include: the breast cancer data set [2], which contains 9 feature attributes and 569 data instances; the Higgs bosons presence data set [3] that contains 28 feature attributes and 11,000,000 data instances (which we only use 11,000 of them); and optical character recognition (OCR) of handwritten digits data set [2] which contains 64 feature attributes and 5620 data instances. We use the centralized SVM as the benchmark. Among these three data sets, the cancer data set is the easiest one: with $50/50$ training and testing, the correct classification ratio is 95%. The Higgs data set is hard to classify because its two classes are highly inseparable: with $50/50$ training and testing, the correct classification ratio is only 70%. The OCR data is easy to be classify, with $50/50$ training and testing, the correct classification ratio is 98%. Cancer data is chosen to compare the performance of our scheme against the bench mark. Higss data is chosen to study the case when the knowledge is hard to learn. OCR data set is chosen to study the vertical partitioned scenario when there are many feature attributes which are highly correlated with each other. Because in this scenario, the distributed learners need to work closely with each other to find the desired information.

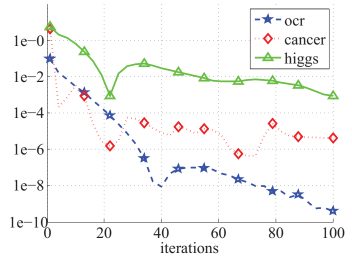
Both our horizontally and vertically partitioned SVM schemes are tested against the three data sets. For the horizontally partitioned scenario, we assume there are 4 learners $M = 4$, and each record is randomly assigned to one learner. For the vertically partitioned case, features are randomly assigned to one of the learners. The slack variable penalty $C = 50$, and the learning speed parameter $\rho = 100$. These two parameters are highly related to the learning performance. If C is set to a high value, then, the primary goal of SVM is to find the hyper-plane that is able to separate the two classes strictly, but it gives a lower priority to the width of the margin. If ρ is set to be high, we put more emphasis on convergence than the max-margin property.

The learning performance of the horizontally partitioned case is shown in Fig. 4. For the 4 learners, we only plot the results at learner 1 because the situations are similar at the other learners. Fig. 4(a) and (b) plot the convergence of z. It is shown that as the iterations increase, z is converging to its optimal value. These two figures could be understood with Fig. 4(e) and (f), to find the improvement of performance as z is converging. Generally speaking, the Higgs data set takes the longest time to converge because the knowledge is hard to discover. If the local learning results are very different, then it may take more effort for them to reach a consensus.

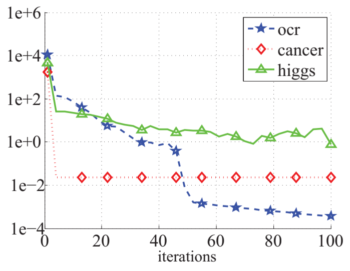
The results on the vertically partitioned data are also illustrated in Fig. 4. It is interesting to see that in Fig. 4(d), among the three data sets, z is converging very fast with the cancer data set, followed by the Higgs data set and finally the OCR data set. There might exist two reasons for this phenomenon: firstly, OCR data has the most number of feature attributes, and hence there are more information to exchange; secondly, features of one record in OCR data set are highly correlated, and thus the distributed learners need to work more closely and intensively to figure out the correlation between their local results. The learning process could be clearly observed in Fig. 4(f) and (g). After a few steps, the classifier gets a very good performance over the cancer data, but for OCR, it takes quite a few iterations for the distributed learners to construct an accurate classifier. Similar situations happens to the nonlinear scheme. Compared with Fig. 4 (b) and (f) we can see that the curve of z has some sudden jumps, however the classification performance curve in (f) doesn't reflect those jumps. It may due to the fact that features assigned to this learner are redundant features. Feature selection could be used to remove the jumps, however, feature selection is also a centralized operation. We may need to design another totally different protocol to achieve distributed feature selection.



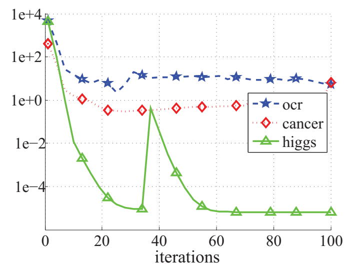
(a) $\|z^{t+1} - z^t\|^2$ linear horizontal



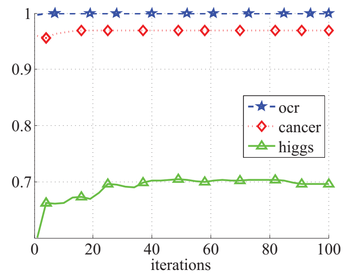
(b) $\|z^{t+1} - z^t\|^2$ nonlinear horizontal



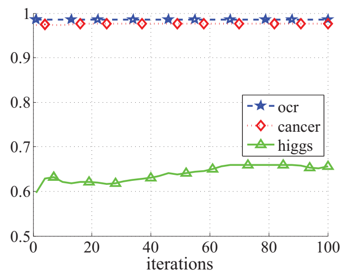
(c) $\|z^{t+1} - z^t\|^2$ linear vertical



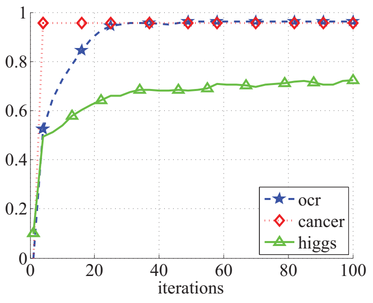
(d) $\|z^{t+1} - z^t\|^2$ nonlinear vertical



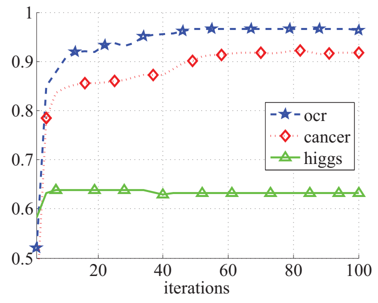
(e) Correct ratio linear horizontal



(f) Correct ratio nonlinear horizontal



(g) Correct ratio linear vertical



(h) Correct ratio nonlinear vertical

Fig. 4: Simulation results with three different data set

VII. Conclusion

In this paper, we studied the problem of collaborative machine learning over distributed training data and proposed to use the data locality property of big data processing framework such as MapReduce to achieve privacy preservation. Generally speaking, the collaborative learning problem is decomposed into subtasks such that each subtask is only related to one share of the training data. With this decomposition, local Mappers are able to work independently to get local training results, which are then summarized by a secure protocol on Reducer. The proposed framework avoids secure operations on Mappers and only use a limited number of cryptographic operations on Reducer to achieve privacy-preservation with an affordable computation overhead. Performance of the proposed scheme is studied via theoretical analysis and extensive experiments over three real-life data sets.

References

[1] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, pages 439–450, New York, NY, USA, 2000. ACM.

[2] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[3] P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nat Commun*, 5, July 2014.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, Jan. 2011.

[5] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[6] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[7] K. Chaudhuri and C. Monteleoni. Privacy-preserving logistic regression. In *NIPS*, pages 289–296, 2008.

[8] K. Chen and L. Liu. Privacy preserving data classification with rotation perturbation. In *ICDM*, pages 589–592, 2005.

[9] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for privacy preserving distributed data mining. *SIGKDD Explor. Newsl.*, 4(2):28–34, Dec. 2002.

[10] C. Cortes and V. Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, Sept. 1995.

[11] Y. Duan, J. Canny, and J. Zhan. P4p: Practical large-scale privacy-preserving distributed computation robust against malicious users. In *Proceedings of the 19th USENIX Conference on Security*, USENIX Security'10, pages 14–14, Berkeley, CA, USA, 2010. USENIX Association.

[12] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and G. Fox. Twister: A runtime for iterative mapreduce. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, pages 810–818, New York, NY, USA, 2010. ACM.

[13] P. K. Fong and J. Weber-Jahnke. Privacy preserving decision tree learning using unrealized data sets. *Knowledge and Data Engineering, IEEE Transactions on*, 24(2):353–364, Feb 2012.

[14] P. K. Fong and J. Weber-Jahnke. Privacy preserving decision tree learning using unrealized data sets. *Knowledge and Data Engineering, IEEE Transactions on*, 24(2):353–364, Feb 2012.

[15] P. A. Forero, A. Cano, and G. B. Giannakis. Consensus-based distributed support vector machines. *J. Mach. Learn. Res.*, 99:1663–1707, August 2010.

[16] G. H. Golub and C. F. Van Loan. *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[17] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Trans. on Knowl. and Data Eng.*, 16(9):1026–1037, Sept. 2004.

[18] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Trans. Knowl. Data Eng.*, 16(9):1026–1037, 2004.

[19] 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*, KDD '06, pages 618–624, New York, NY, USA, 2006. ACM.

[20] Y. Lindell and B. Pinkas. Privacy preserving data mining. *J. Cryptology*, 15(3):177–206, 2002.

[21] O. L. Mangasarian and E. W. Wild. Privacy-preserving classification of horizontally partitioned data via random kernels. In *DMIN'08*, pages 473–479, 2008.

[22] O. L. Mangasarian, E. W. Wild, and G. M. Fung. Privacy-preserving classification of vertically partitioned data via random kernels. *ACM Trans. Knowl. Discov. Data*, 2(3):12:1–12:16, Oct. 2008.

[23] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In *Neural Networks for Signal Processing [1997] VII. Proceedings of the 1997 IEEE Workshop*, pages 276–285, Sep 1997.

[24] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft Research, April 1998.

[25] B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[26] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 639–644, New York, NY, USA, 2002. ACM.

[27] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *KDD*, pages 639–644, 2002.

[28] H. Yu, X. Jiang, and J. Vaidya. Privacy-preserving svm using nonlinear kernels on horizontally partitioned data. In *Proceedings of the 2006 ACM symposium on Applied computing*, SAC '06, pages 603–610, New York, NY, USA, 2006. ACM.

[29] H. Yu, X. Jiang, and J. Vaidya. Privacy-preserving svm using nonlinear kernels on horizontally partitioned data. In *Proceedings of the 2006 ACM Symposium on Applied Computing*, SAC '06, pages 603–610, New York, NY, USA, 2006. ACM.

[30] J. Yuan and S. Yu. Privacy preserving back-propagation neural network learning made practical with cloud computing. *Parallel and Distributed Systems, IEEE Transactions on*, 25(1):212–221, Jan 2014.

[31] J. Z. Zhan and S. Matwin. Privacy-preserving support vector machine classification. *IJIIDS*, 1(3/4):356–385, 2007.