Preserving Model Privacy for Machine Learning in Distributed Systems

Qi Jia*, Linke Guo*, Zhanpeng Jin*, Yuguang Fang†
*Department of Electrical and Computer Engineering, Binghamton University, Binghamton, NY 13902, USA
†Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611, USA
Email: {qjia1, lguo, zjin}@binghamton.edu, fang@ece.ufl.edu

***Abstract*—Machine Learning based data classification is a widely used data mining technique. By learning massive data collected from the real world, data classification helps learners discover hidden data patterns. These hidden data patterns are represented by the learned model in different machine learning schemes. Based on such models, a user can classify whether the new incoming data belongs to an existing class; or, multiple entities may test the similarity of their datasets. However, due to data locality and privacy concerns, it is infeasible for large-scale distributed systems to share each individual's datasets for classifying or testing. On the one hand, the learned model is an entity's private asset and may leak private information, which should be well protected from all other non-collaborative entities. On the other hand, the new incoming data may contain sensitive information which cannot be disclosed directly for classification. To address the above privacy issues, we propose an approach to preserve the model privacy of the data classification and similarity evaluation for distributed systems. With our scheme, neither new data nor learned models are directly revealed during the classification and similarity evaluation procedures. Based on extensive real-world experiments, we have evaluated the privacy preservation, feasibility, and efficiency of the proposed scheme.**

***Index terms*— Machine Learning, Privacy Preservation, Data Classification, Model Evaluation**

I. Introduction

Tremendous amounts of data has surrounded us and affected every aspect of our daily life. To discover hidden data structures from collected data, many data mining techniques have been developed in recent years, which help us figure out useful information from massive messages [1]. As an important analytical method, among all data mining approaches, machine learning based data classification plays a significant role. Generally, data classification mainly consists of two parts, data learning and predicting. During the learning phase, the learning entity applies different methods to learn the dataset and divides the data into various classes according to different data features. Such division can be seen as a learned model which reflects the data structure. Then, in the predicting phase, this model plays as a classifier to classify the incoming test data into the corresponding class. With the growing demands of machine learning techniques, many online machine learning services are also created to remotely handle the learning process for the data owner and provide APIs to predict the new test data, such as BigML, Amazon ML, Google ML, etc [2]. However, although the machine learning and data classification has been making a lot of benefits to our life, it still has some limitations. One major concern is the privacy problem.

The work of L. Guo was partially supported by the National Science Foundation under grants IIS-1722731 and ECCS-1710996. The work of Z. Jin was partially supported by the National Science Foundation (NSF) under grants CNS-1422417 and ECCS-1462473. The work of Y. Fang was partially supported by National Science Foundation (NSF) under grants IIS-1722791.

Many applications of machine learning are related to the sensitive or private data. For instance [3], as shown in Fig. 1, with or without the help of learning services, patient databases containing clinical and genomic data can be learned to construct a supervised model. Then, this learned model can guide medical treatments based on the genotype and background of different patients in the predicting process. Moreover, since different models reflect the structure of different databases, the comparison of such genomic models can be used to realize the medical genetic similarity of various patient groups. Apparently, these applications of data classification and model evaluation can improve the accuracy and efficiency of disease prevention and diagnosis.

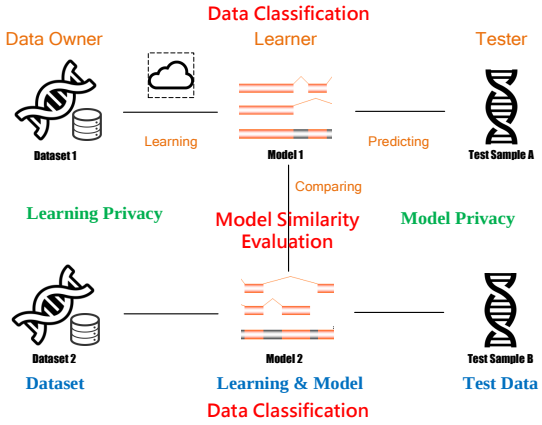


Fig. 1: Data Classification and Model Similarity Evaluation

Unfortunately, the stumbling block is the privacy issue existing in the above classification and evaluation process.

- First, since the **original data** is possessed by each data owner, no matter how the model is used, the data should be kept confidential to other entities. This is a learning privacy problem. Most existing works [4]–[14] focus on solving this problem, where the learning data is protected, but the learned models are published to everyone.
- Second, the **learned model** is valuable and private to the learner, it should be kept secret in the predicting or comparing process. The leakage of a model result in the loss of interests of data owner or even breach the original data. This is a model privacy problem. Without any protection, the predicting results can be utilized to reversely compute the model and even retrieve the original data in recent adversarial studies [15]–[20].
- Third, the **test data** is also private and should be protected properly in the predicting phase. Since the test data are

predicted by the learned model, it also belongs to the model privacy problem. Few works pay the attentions to this part [21]–[24], but their schemes are either inefficient or lack of practical applications.

As a result, it is necessary to preserve the privacy of learning data, learned models and test data among the data classification and model evaluation processes. In this work, we put our efforts to the model privacy issues. We propose a practical and efficient privacy-preserving data classification and model similarity evaluation scheme to address the aforementioned issues. We guarantee that both learned model and the test data are well protected in the predicting phase of data classification, and we invent a new similarity evaluation metric with privacy preservation for the model closeness comparison.

In summary, we have the following major contributions:

- Our proposed privacy-preserving data classification scheme ensures that the learners can conduct data classification successfully without exposing their learned models to the tester, while the testers keep their data in private. The Oblivious Evaluation of Multivariate Polynomial (OMPE) approach is applied to the Support Vector Machine (SVM) decision function to provide the privacy preservation of learned model and test data.
- We propose a privacy-preserving geometric metric to evaluate the closeness of different models. The proposed scheme can privately evaluate model similarities between different learners without exposing their models to each other. Both distance and angle of different model hyperplanes of high dimensional data space are considered for the similarity evaluation.
- Two different privacy levels are elaborated in our scheme for both data classification and model similarity evaluation. The possible attacks and corresponding countermeasures are analyzed in various situations. Our correctness, feasibility, and efficiency are shown by the extensive simulations and experiments on the real-world data.

The rest of this paper is organized as follows. The related works are briefly reviewed in Section II. In Section III, we provide the preliminaries of our scheme. Then, we explain the adversarial model and possible threats in Section IV. After that, our proposed schemes are elaborated in Section V and Section VI for different scenarios. The privacy analysis is discussed in VII. The experimental evaluations are provided in Section VIII. Finally, we conclude this paper in Section IX.

II. Related Work

A. Privacy Preservation in Data Classification

Privacy issues in data classification can be categorized into learning and model privacy. The issues raised in the learning process are about how to protect the learning data and privately compute the learning models. Randomization approaches are applied in the early time. In [4]–[7], they mix the learning data by using random rotation perturbation or random matrix. The learning process is manipulated on the randomized data and the correct learned model can be acquired after the derandomization. Other approaches are based on the cryptographic methods. Several systems [8]–[11] propose solutions by providing the learning process on the encrypted data. The homomorphic cryptosystem is applied so that the optimization problems can still be solved by the homomorphic properties in the ciphertext. Liu *et. al.* apply the secret sharing scheme to protect the privacy by requiring all users' communication and cooperation in [6]. Recently, most research works move further steps to the distributed data systems. The proposed approaches such as [25]–[29] guarantee the universal optimization value for the learning target over the whole distributed system, while will not require the actual data information for each individual data owner. The privacy concerns in such systems usually concentrate on the passing gradient values instead of the real data [12]–[14]. In [25], Xie *et. al.* provide an asynchronous update model to solve the secure learning problem for the general class of multi-task learning (MTL). Xu *et. al.* [30]–[32] analyze the distributed privacy preserving SVMs in both the horizontally and vertically partitioned data by modifying the Alternating Direction Method of Multiplier (ADMM) schemes. The assumption behind all these works is the learning data is private but learned models can be publicly known.

However, the learned model should also be protected. Fredrikson *et. al.* [16] analyze the feasibility to retrieve original learning data from the known model. They successfully reconstruct the face images from the Neural Network face recognition model. In [17], Tramèr *et. al.* implement the model stealing to several popular learning algorithms and show that only the test results can be used to recover the model information. To prevent such privacy leakages, Rahulamathavan *et. al.* [22] propose a scheme to transform the SVM decision function into an encrypted form by the Paillier cryptosystem. The classification sample is also encrypted. All the computations are operated on the ciphertext and only the tester who holds the private key can decrypt the classified result. The other related work to our approach is [24]. Raphael *et. al.* introduced the privacy-preserving classification schemes over hyperplane decision, Naive Bayes, and decision trees classifiers. They implement several homomorphic cryptosystems in different steps of data classification to protect the privacy of learned model and tester inputs. However, the homomorphic cryptosystems would introduce complicated encryption procedures. Compared to their approach, we apply a uniform OMPE method without additional encryption for the classifications and provide the similarity evaluation to different learned models.

B. Privacy Preservation in Data Similarity Evaluation

Our privacy-preserving model evaluation is close to the matching system, which has been applied in finding the best-matched pair of users that have most similar interests or behaviors on social networks. Many elegant and efficient matching systems have been proposed. In [33], [34], they propose coarse-grained privacy-preserving matching systems. In their descriptions, the matching attributes are derived from a public set so that the matching problems become to the Private Set Intersection problems. In [35], Zhang *et. al.* propose a fine-grained matching system, where all attributes are graded in different values. However, the grading process is not clear in this paper. Users can input any values for the matching purpose, and this arbitrary inputs could lead to a totally mismatched result. In our proposed solutions, the evaluation is based on the learned models, which is the reflection of inherent behaviors as an objective comparison result.

III. Preliminaries

A. Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning scheme that divides the data space into two different

parts by a binary classifier based on the learning data [36]. Assuming data $\mathbf{x} \in \mathbb{R}^n$ is an n dimensional vector, and it has a label $y \in \{+1, -1\}$ indicating its class. The learning process is to find a hyperplane that divides the data space with the largest margin distance between different data label groups.

In a linear classification problem, an unlabeled data sample $\mathbf{t}$ can be categorized into label $y_\mathbf{t} = +1$ (if $\mathbf{w}^T\mathbf{t} + b > 0$) or $y_\mathbf{t} = -1$ (if $\mathbf{w}^T\mathbf{t} + b < 0$) with a linear hyperplane. The learned decision function $d(\mathbf{t})$ can be summarized as

$$d(\mathbf{t}) = \mathbf{w}^T\mathbf{t} + b = \sum_{s \in S} \alpha_s y_s \mathbf{x}_s^T \mathbf{t} + b \tag{1}$$

where $\mathbf{w}$ and b are the optimized model parameters. S is the set of support vectors, $\mathbf{x}_s, 1 \le s \le |S|$ are the support vectors, and $y_s, 1 \le s \le |S|$ are the corresponding labels. Here, $\alpha_s, 1 \le s \le |S|$ are the Lagrangian multiplier parameters. Then, a class label will be assigned to the test sample $\mathbf{t}$ by finding the sign of the decision function output as follow,

$$D(\mathbf{t}) = \mathbf{sign}(d(\mathbf{t})) = \mathbf{sign}(\mathbf{w}^T\mathbf{t} + b) = \mathbf{sign}(\sum_{s \in S} \alpha_s y_s \mathbf{x}_s^T \mathbf{t} + b).$$

In the nonlinear classification, data is nonlinearly distributed and it can hardly find a linear hyperplane to divide the data in the original data space. Therefore, kernel functions [36] are used to map the low dimensional data to a higher dimensional space, where the linear SVM method can be applied. For the nonlinear SVM decision function, the dot product $\mathbf{x}^T\mathbf{t}$ is substituted by kernel functions. The change is

$$\mathbf{x}^T\mathbf{t} \Rightarrow K(\mathbf{x}, \mathbf{t}) = < \varphi(\mathbf{x}), \varphi(\mathbf{t}) >$$

where $\varphi : \mathbb{R}^n \to \mathbb{R}^{n'}, n < n'$ is a high dimensional mapping. Then, the nonlinear classification function can be expressed as

$$\begin{aligned} D(\mathbf{t}) &= \mathbf{sign}(d(\mathbf{t})) = \mathbf{sign}(< \varphi(\mathbf{w}), \varphi(\mathbf{t}) > + b) \\ &= \mathbf{sign}(\sum_{s \in S} \alpha_s y_s K(\mathbf{x}_s, \mathbf{t}) + b) \end{aligned}$$

B. Oblivious Transfer Protocol

Oblivious transfer is one important cryptographic primitive protocol to provide a secure scheme for data transfer among different parties. Normally, there are two parties in the protocol [37]–[39], receiver and sender, for transmitting messages. By executing this protocol, the receiver only gets one part of the information from the sender without knowing other parts, while the sender does not know which part is obtained by the receiver. Generally, the protocol is developed in three steps:

1) 1-out-of-2 protocol: The sender has two messages $\{m_1, m_2\}$. The receiver has one bit b. The receiver wants to get message m_b without exposing b to the sender. While the sender can ensure that only one of two messages is delivered to the receiver.
2) 1-out-of-n protocol: It is a generalization of 1-out-of-2 protocol. The receiver has an index σ and wishes to get the σth message among the message set $\{m_1, ..., m_n\}$, while the sender can ensure only one of the messages is sent without knowing the index σ.
3) k-out-of-n protocol: In this case, the receiver has a group of indices $\{\sigma_1, ..., \sigma_k\}$. After this protocol, the receiver only gets the corresponding messages $\{m_{\sigma_1}, ..., m_{\sigma_k}\}$ from $\{m_1, ..., m_n\}$, while the sender will not know about the indices.

C. Oblivious Evaluation of Multivariate Polynomials

Oblivious Evaluation of Multivariate Polynomials (OMPE) is introduced in [40]. It is a secure multi-party computation protocol that can securely compute the multivariate polynomial functions. The proposed scheme assumes that the sender has r variates with d-degree polynomial $P(\cdot)$, and the receiver has the input vector $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_r)$. The protocol can ensure that the receiver can get the polynomial computation result $P(\boldsymbol{\alpha})$ without exposing $\boldsymbol{\alpha}$ to the sender, while the polynomial $P(\cdot)$ is still kept secret from the receiver. The OMPE procedures run as follows,

1) The sender generates a random univariate masking polynomial $M(x)$ with degree sd, where s is a security parameter and $M(0) = 0$. Then, the sender hides real polynomial $P(\cdot)$ by defining a $(r+1)$ variates polynomial $Q(x, \mathbf{y}) = M(x) + P(\mathbf{y})$.
2) The receiver generates r random univariate masking polynomials $S_i(\cdot)$ and ensures that $S_i(0) = \alpha_i$. Then, the receiver collects the $S_i(\cdot)$ and defines a tuple $\mathbf{S}(x) = (S_1(x), ..., S_r(x))$.
3) The receiver generates $N = nm$ random inputs $x_1, ..., x_N$, where $n = sd + 1$ and m is a secure random number. The receiver also selects n positions among the inputs to compute $\mathbf{y}_i = \mathbf{S}(x_i)$ as covers. For other positions, $\mathbf{y}$ are just randomly selected.
4) The receiver sends N pairs value $\{x_i, \mathbf{y}_i\}$ to the sender, the sender calculates the corresponding values $Q(x_i, \mathbf{y}_i)$.
5) The receiver and sender execute the n-out-of-N protocol, and the receiver gets n selected position values as $R(x_i) = Q(x_i, \mathbf{S}(x_i))$.
6) The receiver interpolates the acquired values to get $R(x)$, and the function result is $P(\boldsymbol{\alpha}) = Q(0, \boldsymbol{\alpha}) = Q(0, \mathbf{S}(0)) = R(0)$.

IV. Adversarial Model

In this section, we present the basic privacy assumptions in our design and explain possible privacy breaches during the classification process.

A. Adversarial Assumptions

To illustrate the privacy challenges in our scheme, we put forward two assumptions for this system. First, we assume all users in our system are honest-but-curios, i.e., they will try to learn more information than allowed by inferring from the results, but they will honestly follow the schemes. Second, we assume the collusion may happen among learners and/or testers. A learner/tester may collude with other learner/tester to speculate more messages on one tester/learner. However, learners should not collude with the tester. Based on such assumptions, we bring out two different privacy threats in the proposed scenario.

B. Privacy Breaches

1) Explicit Leakage: Generally, the normal classification process requires the participation of both learned model and test data. For example, the test data will be sent to the learner for predicting its class, which violates the privacy requirement of the test data. Therefore, the test data or learned model should not be directly exposed to the other parties during the predicting phase, which means the model parameters should be kept secret on the learner side, while test data vectors are in private on the tester side.

2) Implicit Leakage: The information should be prevented from indirectly obtaining of learner or tester. In other words, after the classification, neither learner nor tester should retrieve the test data or model parameters from their classification results. On the tester side, the test data are different in each classification request, such that the learner can hardly retrieve the test data just from one specific result. However, on the learner side, the learned model is the same in every classification requests. Multiple test results could be utilized to retrieve the original model. Based on the above models, we consider three different attack strategies in this work.

C. Attacking Strategy

1) Model Rebuild Attack: Obtaining the accurate output values of the classification results, this attack can solve a group of equations from the test results to reconstruct the model. For example, for an n dimensional classification model, only $n+1$ singularity equations are sufficient to reconstruct all the model parameters. As shown in Fig. 2, for a visual 2-D data space, only knowing three points and corresponding distances is enough to rebuild a decision function by finding the common tangent.

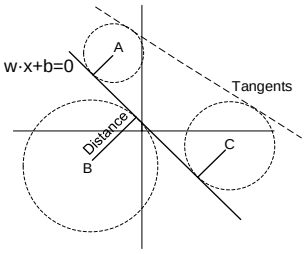


Fig. 2: Model Rebuild Attack

2) Data Fitting Attack: Knowing the distribution of the classification results, this attack can use data fitting algorithms to retrieve the original information. As shown in Fig. 3, the Least Square (LS) approach is applied to a bunch of data points. These points have been randomly biased from the correct outputs. However, the LS attack can still reconstruct a similar division to the original model. Comparing with the model rebuild attack, it computes a statistical average such that the randomization influence of an individual point is reduced.

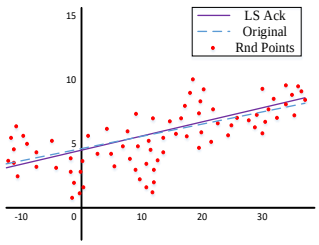


Fig. 3: Data Fitting Attack

3) Retraining Attack: Having the labels of the classification results, this attack directly applies the learning method to the test data to obtain a new model. As long as the test data covers all the output classes, the rebuilt model can be similar to the original model with increasing test results. Due to the exposure of the final output labels, this attack is inevitable from the predicting process. However, the difficulty of retraining attack equals to have a new learning process, which requires more classification results than the model rebuild or data fitting attacks.

V. Privacy Preserving Data Classification

In this section, we propose a privacy-preserving data classification scheme for computing the classes of the tester' samples. Assuming Alice is the learner who holds the learning dataset and has derived a classifier, while Bob is the tester who wants to figure out which class his test data sample belongs to. As shown in Fig. 4, the black curve in (a) can be reviewed as the learned model of Alice, it divides the data space into two different classes. The circle in Fig. 4(b) represents Bob's sample which is unlabeled. To acquire the class label for the sample, our proposed scheme guarantees that Bob can get the data classification result, while it also achieves the learned models' and classification samples' privacy preservation during the computation. Basically, we firstly analyze the SVM decision functions. Then we apply the OMPE protocol to securely compute the classification result. Finally, the class can be classified from the sign of decision function result. To elaborate our scheme, both linear and nonlinear data classification problems are described.

(a) Learned Pattern (b) Test Data

Fig. 4: Data Classification

A. Privacy-preserving Linear Data Classification

In the linear data classification problem, the data is located linearly in the data space. Alice has learned a linear classifier $d(\mathbf{t})$ from her dataset, and Bob wants to get the classification result for his sample $\tilde{\mathbf{t}}$. In this scenario, the classifier is a linear hyperplane that divides the data space into two different parts. To protect its privacy, we can start the analysis with the linear SVM decision function.

$$d(\mathbf{t}) = \mathbf{w}^T\mathbf{t} + b = \sum_{s\in S} \alpha_s y_s \mathbf{x}_s^T \mathbf{t} + b$$
$$= \sum_{s\in S} \alpha_s y_s x_{s1} t_1 + ... + \sum_{s\in S} \alpha_s y_s x_{sn} t_n + b$$

It is an n variates polynomial with degree 1. To privately get the classification result of the decision function, Alice and Bob execute the following steps.

1) Trained Model Randomization: Alice has the learned model $d(\mathbf{t})$ as an n variates polynomial with degree 1. Bob initiates the request with Alice for privately classifying his test data. Then, Alice generates a random univariate polynomial $h(u)$ with the degree of q, where q is a security random parameter. She also makes sure that $h(0) = 0$, such that,

$$h(u) = \sum_{1\le j\le q} a_j u^j$$

where the $a_j \in \mathbb{R}$ is a random coefficient of u^j. After generating $h(u)$, Alice selects a random number $r_a > 0, r_a \in \mathbb{R}$ to amplify her decision function as $d'(\mathbf{t}) = r_a d(\mathbf{t})$. Then, she combines the random polynomial and decision function together to reach an $(n+1)$ variates polynomial as follows,

$$\begin{aligned} A(u, \mathbf{t}) &= h(u) + d'(\mathbf{t}) \\ &= \sum_{1 \le j \le q} a_j u^j + r_a \sum_{s \in S} (\alpha_s y_s \mathbf{x}^T \mathbf{t} + b) \\ &= \sum_{1 \le j \le q} a_j u^j + (r_a \sum_{s \in S} \alpha_s y_s x_{s1}) t_1 + ... \\ &+ (r_a \sum_{s \in S} \alpha_s y_s x_{sn}) t_n + r_a b \end{aligned} \tag{2}$$

which forms an $(n+1)$ variates polynomial of degree q with inputs $(u, t_1, ..., t_n)$.

2) Classification Data Randomization: Given test sample $\tilde{\mathbf{t}} = \{\tilde{t}_1, ..., \tilde{t}_n\}$, Bob generates n random univariate polynomials of degree q to hide each component. Let the polynomials be $\{g_i(\cdot)\}_{1 \le i \le n}$, Bob makes sure that $g_i(0) = \tilde{t}_i$ as well,

$$g_i(v) = \sum_{1 \le j \le q} b_j v^j + \tilde{t}_i$$

where b_j are the random numbers from $\mathbb{R}$. After generating parameters, Bob combines the polynomials together to get a vector $\mathbf{G}(v)$, such that,

$$\mathbf{G}(v) = (g_1(v), ..., g_n(v))$$

Then, Bob computes a global number $M = mk$, where $m = q+1$ and k is a secret random number. He generates M non-zero random numbers $v_1, ..., v_M \in \mathbb{R}$, and selects m indices from these random numbers as a subset I. Assume the selected indices are $I = \{\sigma_1, ..., \sigma_m\}$, $1 \le \sigma_1 < ... < \sigma_m \le M$, for these selected numbers, Bob computes $\mathbf{z}_{\sigma_i} := \mathbf{G}(v_{\sigma_i}), \sigma_i \in I$. In this way, m selected terms are hided in all M numbers with partial information of $\mathbf{G}$. For any other indices $j \notin I$, $\mathbf{z}_j$ are randomly selected from $\mathbb{R}^n$ as disguise values. Finally, Bob sends these M pair values $\{(v_i, \mathbf{z}_i)\}_{1 \le i \le M}$ to Alice.

3) Classification Result Retrieval: After receiving M pairs from Bob, Alice computes $A(v_i, \mathbf{z}_i)$ in Eq. (2) for all received values. Then, Alice and Bob can execute an m-out-of-M oblivious transfer protocol. Bob can only get the value of his selected indexes, while Alice does not know which values are delivered to Bob. Therefore, Bob only learns the value $\{A(v_i, \mathbf{z}_i), i \in I\}$, and Alice learns nothing because she cannot reveal the real values of t_i in feasible polynomial time [40]. If we examine the received values of Bob,

$$A(v_i, \mathbf{z}_i) = A(v_i, \mathbf{G}(v_i)) = h(v_i) + d'(\mathbf{G}(v_i)), i \in I$$

it is a univariate q-degree polynomial of variate v. Bob redefines this function as $B(v)$, $B(v) = A(v, \mathbf{G}(v))$. As a q-degree univariate polynomial, it can be reconstructed by $(q+1)$ pair values of inputs and outputs. Since we have chosen $|I| = m = q+1$ pairs of value $\{A(v_i, \mathbf{z}_i), i \in I\}$. The Lagrange interpolation can be implemented to reconstruct $B(v)$. The interpolation can be done as follow,

$$\begin{aligned} B(v) &= \sum_{j=1}^{m} B_j(v) \\ B_j(v) &= B(v_j) \prod_{i=1, i \ne j}^{m} \frac{v - v_i}{v_j - v_i} \ . \end{aligned} \tag{3}$$

Once Bob gets $B(v)$ from the interpolation, he can get the $d'(\tilde{\mathbf{t}})$ by computing the $B(0)$. This is because

$$\begin{aligned} B(0) &= A(0, \mathbf{G}(0)) = h(0) + d'(\mathbf{G}(0))) \\ &= d'(\tilde{t}_1, ..., \tilde{t}_n) = d'(\tilde{\mathbf{t}}) \ . \end{aligned}$$

Finally, to decide which class is correct for the classification sample $\tilde{\mathbf{t}}$, Bob just needs to get the sign of $d(\tilde{\mathbf{t}})$. Since it has been amplified with a positive random number r_a, $d'(\mathbf{t})$ keeps the same sign as $d(\mathbf{t})$. So, the classification result is

$$D'(\tilde{\mathbf{t}}) = \mathbf{sign}(d'(\tilde{\mathbf{t}})) = \mathbf{sign}(r_a d(\tilde{\mathbf{t}})) = \mathbf{sign}(d(\tilde{\mathbf{t}})) = D(\tilde{\mathbf{t}})$$

Hence, Bob gets the classification result without exposing his values, while Alice keeps her decision function undisclosed.

B. Privacy-preserving Nonlinear Data Classification

In nonlinear data classification scenario, the data is distributed nonlinearly in the data space and it is impossible to divide the original data with a linear hyperplane. In such case, the decision function is improved by the kernel methods,

$$d(\mathbf{t}) = \sum_{s \in S} \alpha_s y_s K(\mathbf{x}_s, \mathbf{t}) + b$$

In this decision function, the kernel function is used to map the lower dimensional vectors to a higher dimension. Various kernel functions can be applied to different data distribution. Here we list three most popular kernel functions:

- Polynomial kernel: $K(\mathbf{x}, \mathbf{y}) = (a_0 \mathbf{x}^T \mathbf{y} + b_0)^p$
- Radial Basis Function Kernel: $K(\mathbf{x}, \mathbf{y}) = c_0 e^{||\mathbf{x}-\mathbf{y}||^2}$
- Sigmoid Kernel: $K(\mathbf{x}, \mathbf{y}) = \mathbf{tanh}(\mathbf{x}^T \mathbf{y} + d_0)$

Among them, if the polynomial kernel is applied, the nonlinear decision function can still be seen as an n variates polynomial function of degree p. For the radial basis function or sigmoid kernel, we can apply the Taylor Expansion to transform them into polynomials. Therefore, the corresponding nonlinear decision functions are reformed as below,

- Polynomial kernel:

$$d(\mathbf{t}) = \sum_{s \in S} \alpha_s y_s (\mathbf{x}^T \mathbf{t})^p + b$$

- Radial Basis Function Kernel:

$$d(\mathbf{t}) = \sum_{s \in S} \alpha_s y_s \sum_{i=0}^{\infty} \frac{(\mathbf{x} - \mathbf{t})^{2i}}{i!} + b \tag{4}$$

- Sigmoid Kernel:

$$d(\mathbf{t}) = \sum_{s \in S} \alpha_s y_s \sum_{i=0}^{\infty} [\frac{B_{2i} 4^i (4^i - 1)}{2i!} (\mathbf{x}^T \mathbf{t})^{2i-1}] + b$$

In real applications, we can use a limited number p to approximate the infinity. Actually, in most cases, only a small number of p is sufficient to obtain enough classification accuracy. In doing so, no matter what kind of kernel function is used in the nonlinear SVM, it can always be transformed into p-degree polynomials. For the nonlinear polynomial kernel function, if we set $a_0 = 1$, and $b_0 = 0$, the data classification processes should be updated to the nonlinear scenario and the corresponding decision function is,

$$d(\mathbf{t}) = \sum_{s\in S}\alpha_s y_s(\mathbf{x}^T\mathbf{t})^p + b$$
$$= \sum_{s\in S}\alpha_s y_s(x_1t_1 + ... + x_nt_n)^p + b$$
$$= \sum_{s\in S}\alpha_s y_s(\sum_{k_1+...+k_n=p}\binom{p}{k_1,...,k_n}\prod_{1\le i\le n}x_i^{k_i}t_i^{k_i}) + b =$$
$$\sum_{k_1+...+k_n=p}\{[\sum_{s\in S}(\alpha_s y_s\binom{p}{k_1,...,k_n}\prod_{1\le i\le n}x_{si}^{k_i})][\prod_{1\le i\le n}t_i^{k_i}]\} + b$$

In this case, $d(\mathbf{t})$ is an $n' = \binom{n+p-1}{n-1}$ variates polynomial, and input terms $\prod_{1\le i\le n}t_i^{k_i}$ are the combination multiplications of original variates t_i. If we treat each term as a variate τ_j, $\tau_j = \prod_{1\le i\le n}t_i^{k_i}, 1 \le i \le n, 1 \le j \le n'$, we can rewrite $d(\mathbf{t})$ as $d(\boldsymbol{\tau})$, where $d(\boldsymbol{\tau})$ is just an n' variates p-degree polynomial.

For the RBF kernel function, if we set $c_0 = 1$, the corresponding decision function in polynomial form is,

$$d(\mathbf{t}) = \sum_{s\in S}\alpha_s y_s\sum_{i=0}^{p}\frac{(\mathbf{x}-\mathbf{t})^{2i}}{i!} + b$$
$$= \sum_{s\in S}\alpha_s y_s\sum_{i=0}^{p}\frac{1}{i!}(\mathbf{x}^2 + \mathbf{t}^2 - 2\mathbf{x}\mathbf{t})^i + b$$

Since $\mathbf{x}^2$ and $\mathbf{t}^2$ are the norm squares of the learner and tester, we can let a constant $c = \mathbf{x}^2 + \mathbf{t}^2$ be a shared public value. This value cannot be used to retrieve the correct value of $\mathbf{x}$ and $\mathbf{v}$. Then, the above function can be written as,

$$d(\mathbf{t}) = \sum_{s\in S}\alpha_s y_s\sum_{i=0}^{p}[\sum_{k_1+...+k_n=i}\binom{i}{k_1,...,k_n}\prod_{1\le j\le n}x_j^{k_j}t_j^{k_j}] + b$$
$$= \sum_{i=0}^{p}\sum_{k_1+...+k_n=i}\{[\sum_{s\in S}(\alpha_s y_s\binom{i}{k_1,...,k_n}\prod_{1\le j\le n}x_{sj}^{k_j})]$$
$$[\prod_{1\le j\le n}t_j^{k_j}]\} + b$$

Compared to the polynomial kernel, the RBF kernel function is a summation of multiple polynomial kernels. Theoretically, p controls the approximate performance of the Taylor transformation. However, larger p will introduce the higher degree polynomials, which causes more computation loads. Fortunately, in the real classification tasks, the data is normalized to a range that close to the $\mathbf{0}$ point, such that smaller p can perform a sufficient approximation.

Based on such polynomial transformation, we can apply the OMPE approach to this transformed polynomial to achieve the secure computation. In the learned model randomization step, instead of original $(n+1)$ variates, Alice computes an $(n'+1)$ variates with pq-degree polynomial $A(u, \boldsymbol{\tau})$,

$$A(u, \boldsymbol{\tau}) = h(u) + d'(\boldsymbol{\tau})$$

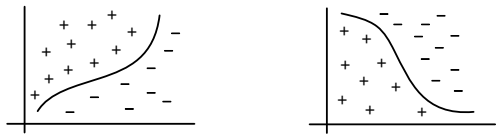
where $h(u)$ is a pq-degree random univariate polynomial with zero constant term. Then, in the data randomization step, Bob transforms his $\tilde{\mathbf{t}}$ to $\tilde{\boldsymbol{\tau}}$, and $\mathbf{G}(v)$ is changed to an n' terms vector, $\mathbf{G}(v) = (g_1(v), ..., g_{n'}(v))$. The $g_i(v)$ is still generated randomly by Bob to hide his data. In the retrieval part, m equals to $(pq + 1)$. As shown in Eq. (3), to interpolate the function $B(v)$, $(pq + 1)$ different value pairs are obliviously transmitted between Alice and Bob. Finally, Bob still gets the classification result by acquiring $d'(\tilde{\boldsymbol{\tau}})$ as,

$$B(0) = A(0, \mathbf{G}(0)) = h(0) + d'(\mathbf{G}(0)))$$
$$= d'(\tilde{\tau}_1, ..., \tilde{\tau}_{n'}) = d'(\tilde{\boldsymbol{\tau}})$$

Therefore, for the nonlinear case, Bob's data can still be securely classified in our nonlinear privacy-preserving data classification scheme. During the whole process, both Alice's learned model and Bob's classification sample are protected.

VI. Privacy Preserving Model Similarity Evaluation

As we mentioned earlier, model similarity evaluation is to calculate the similarities among different models. As shown in the Fig. 5, since learner has different learning data, their learned models could be different from each other. In order to compare two different models, we propose a privacy-preserving model similarity evaluation scheme. To evaluate the similarity, the first step is to find a metric that can properly represent the closeness of different learned models. In our scheme, we combine both factors of the direction and position of learned models as a metric. Then, we explain how to compute this metric with the privacy preservation of different learners. Similarly, our scheme will be elaborated in both linear and nonlinear scenarios.



(a) Learned Model 1 (b) Learned Model 2

Fig. 5: Model Similarity Evaluation

A. Similarity Metric

1) Cosine Similarity: Considering the geometry concepts of SVM, the decision function $d(\mathbf{t})$ is a hyperplane that divides the data space $\mathbb{R}^n$ for classification. Different decision functions will give various hyperplanes that dividing the data space in different directions and positions. Therefore, the model similarity evaluation can be seen as the problem of comparing different hyperplanes. The intuitive way of comparing two high dimensional hyperplanes is to measure the included angle of them. If the data space is unlimited, the comparison can be done by calculating the cosine value of two hyperplanes' included angle. This cosine value can also be calculated by the normal vectors. If $\mathbf{v}$ and $\mathbf{w}$ are the coefficient vectors of the high dimensional decision functions, the cosine value, or called cosine similarity, is calculated as follows,

$$\cos\theta = \frac{\mathbf{v}\cdot\mathbf{w}}{\sqrt{||\mathbf{v}||^2||\mathbf{w}||^2}}$$

However, only comparing the hyperplanes by the angle could be far away from the real similarity of data models. In particular, the data will not be distributed in the infinite space. Normally, they are limited in a certain area of the $n(n')$ dimensional data space. Two hyperplanes may have different angles and locations in the data space. As shown in Fig. 6, the red line is located at the upper region of the data space

and has a relatively horizontal direction, while the blue line is located at the central region of the data space and has a relatively vertical direction. Only comparing their directions cannot truly reflect the real closeness. For this reason, the decision function of the learned model should be considered as bounded hyperplane with different direction angle and boundary points in the limited data space. So, we need to find a metric that can give an assessment of the closeness of two high dimensional bounded hyperplanes instead of two unbounded ones.

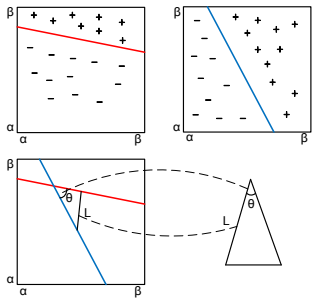


Fig. 6: Bounded Data Space and Isosceles Triangle Construction

2) Isosceles Triangle Area: To protect the bounded planes as well as give a reasonable metric for the closeness, we propose a novel approach to privately compute the closeness of two bounded hyperplanes with the combination of the centroid distance and cosine similarity. The included angle should be kept as a reflection of the directions. To reflect the distance of two bounded planes, we choose to compute their centroid distance. Here, We assume the included angle is θ and the centroid distance is L. By combining these two elements, as shown in Fig. fig:bds, we can construct an isosceles triangle with vertex angle equals to θ and legs equal to L. Hence, we can get the area of this isosceles triangle $T = \frac{1}{2}L^2 \sin\theta$, where $\sin\theta = \sqrt{1-\cos^2\theta}$. Apparently, the area T is related to the sine value of θ and the square value of L, which reflects how close the two bounded plane is in both views of direction and position. To avoid the square roots, we can compute the square of the area as $T^2 = \frac{1}{4}L^4 \sin^2\theta$. However, two extreme situations should be considered. One is the two hyperplanes are parallel, the other one is their centroid locate at the same position. In these cases, the area becomes to null and it is impossible to differentiate which one causes the null area. So, to avoid this problem, we can add two small public constant values as angle $\theta_0 << 90°$ and distance L_0 on the original terms. Then, the square area value becomes as follows,

$$T^2 = \frac{1}{4}(L^4 + L_0^4)(\sin^2\theta + \sin^2\theta_0) \tag{5}$$

Note that only when the two bounded planes are parallel and the centroid are coincident, the area T can get the smallest value $\frac{1}{2}L_0^2 \sin\theta_0$. After acquiring the T^2, T can be computed by computing its square root. This area value T can be an evaluation of learned models. Smaller square area value means two different models are closer, and vice versa. Notice that even we analysis this metric for similarity from the linear decision function and geometric way, it still can be a reflection of the nonlinear situation by kernel method. The nonlinear decision function can be mapped to a higher dimensional data space as a linear hyperplane, and still the centroids and cosine similarity can be found in the higher dimensional space.

B. Privacy-preserving Linear Model Similarity Evaluation

We continue to use Alice and Bob as an example to compare their learned models. In the linear distributed data scenario, their decision functions are

$$d_A(\mathbf{t}) = \mathbf{w}_A^T\mathbf{t} + b_A, \qquad d_B(\mathbf{t}) = \mathbf{w}_B^T\mathbf{t} + b_B$$

where $\mathbf{w}_A \in \mathbb{R}^n := (w_{A_1}, ..., w_{A_n})$, and $\mathbf{w}_B \in \mathbb{R}^n := (w_{B_1}, ..., w_{B_n})$.

1) Angle and Distance Computation: We assume the data space is limited within $[\alpha, \beta]$. The boundary points of Alice or Bob can be found by solving all feasible solutions of $\mathbf{w}^T\mathbf{t} + b = 0, \alpha \le t_i \le \beta$. As shown in Eq. (6), it should treat one dimension value as a variable at each time, and the other dimensions should be arranged in different combinations of α and β. So, for each variate, there are 2^{n-1} equations to be solved and all the feasible solutions are the boundary points. For example, if we treat t_1 as a variable u, the equations are

$$\begin{cases} w_1u + w_2\alpha + ... + w_n\alpha + b = 0 \\ w_1u + w_2\alpha + ... + w_n\beta + b = 0 \\ ... \\ w_1u + w_2\beta + ... + w_n\alpha + b = 0 \\ ... \\ w_1u + w_2\beta + ... + w_n\beta + b = 0 \end{cases} \tag{6}$$

All the feasible u in these equations, where $\alpha \le u \le \beta$, can make up the first dimension boundary points with the corresponding α and β. After this computations, we suppose $\{\mathbf{m}_{A_1}, ..., \mathbf{m}_{A_\gamma}\}$ are the boundary points of Alice and $\{\mathbf{m}_{B_1}, ..., \mathbf{m}_{B_\delta}\}$ are the boundary points of Bob, where γ, δ are the numbers of boundary points. Then, the bounded plane's centroid $\mathbf{m}_A$ of Alice and $\mathbf{m}_B$ of Bob are

$$\mathbf{m}_A = \frac{\sum_{i=1}^{\gamma}\mathbf{m}_{A_i}}{\gamma} \qquad \mathbf{m}_B = \frac{\sum_{i=1}^{\delta}\mathbf{m}_{B_i}}{\delta}$$

The Euclidean distance L of these two centroids is

$$L = \sqrt{|\mathbf{m}_A|^2 + |\mathbf{m}_B|^2 - 2\mathbf{m}_A \cdot \mathbf{m}_B}$$

Based on the cosine similarity and centroid distance, the square area value for Eq. (5) can be written as follows,

$$\begin{aligned} T^2 &= \frac{1}{4}(L^4 + L_0^4)(\sin^2\theta + \sin^2\theta_0) \\ &= \frac{1}{4}[(|\mathbf{m}_A|^2 + |\mathbf{m}_B|^2 - 2\mathbf{m}_A \cdot \mathbf{m}_B)^2 + L_0^4] \\ &\quad [(1 - \frac{(\mathbf{w}_A \cdot \mathbf{w}_B)^2}{|\mathbf{w}_A|^2|\mathbf{w}_B|^2}) + \sin\theta_0^2] \end{aligned} \tag{7}$$

2) Area Computation: As so far, the privacy-preserving similarity evaluation problem has been transformed into a problem on privately computing the square area value. We continue to apply the OMPE protocol to achieve the design objective. However, instead of directly expanding the Eq. (7) as a multivariate polynomial, we can simplify the computations.

First, Bob can send $|\mathbf{m}_B|^2$ and $|\mathbf{w}_B|^2$ to Alice directly. Since these two terms are the vector squares, Alice cannot reveal any dimension's value of them. To decrease the number of variates, instead of computing the whole square area value as a $2n$ variates polynomial for $\mathbf{m}_A$ and $\mathbf{w}_B$, Bob computes $\mathbf{m}_A \cdot \mathbf{m}_B$ and $\mathbf{w}_A \cdot \mathbf{w}_B$ as two input variates. By similarly

running the OMPE protocol, Bob will get the amplified values, $r_{am}(\mathbf{m}_A \cdot \mathbf{m}_B)$ and $r_{aw}(\mathbf{w}_A \cdot \mathbf{w}_B)$. In these values, the random amplifiers r_{am} and r_{aw} are used to prevent Bob to retrieve the real polynomial coefficients of Alice. However, one exception is the value $\mathbf{w}_A \cdot \mathbf{w}_B$ could be zero so that Bob will know Alice is vertical with his plane. He can get the centroid distance by deduce the final result and find out the plane of Alice. To avoid this problem, Alice adds another random value $r_b \in \mathbb{R}$ on the hidden polynomial, such that $d'(\mathbf{t}) = r_{aw}d(\mathbf{t}) + r_b$. Without knowing this additional value, Bob cannot retrieve anything from his obtained values.

Then, to simplify the whole function, we let $c_1 = ||\mathbf{m}_A||^2 + ||\mathbf{m}_B||^2$, $c_2 = L_0^4$, $c_3 = (||\mathbf{w}_A||^2||\mathbf{w}_B||^2)^{-1}$, $c_4 = 1 + \sin\theta_0^2$, and let $d_1 = r_{am}^{-1}$, $d_2 = r_{aw}^{-1}$, $d_3 = -r_b$ be the constants. In addition, Alice lets $x_1 = r_{am}(\mathbf{m}_A \cdot \mathbf{m}_B)$, $x_2 = r_{aw}\mathbf{w}_A \cdot \mathbf{w}_B + r_b$ be the variates delivered by Bob. The square area value can be calculated with following polynomial of x_1 and x_2,

$$T^2(x_1, x_2) = \frac{1}{4}[(c_1 - 2d_1x_1)^2 + c_2][c_4 - c_3d_2(d_3 + x_2)^2] \tag{8}$$

where the degree is 4. This is a simple two-variate polynomial function. All the constants $c_i, i = 1, 2, 3, 4, d_j, j = 1, 2, 3$ are only known by Alice and the variates x_1, x_2 are just delivered form Bob. Applying the OMPE approach again, Alice and Bob can simply get square area value from this simplified function. Finally, Bob can get T as the similarity evaluation result, while all the privacy of Alice and Bob are protected.

C. Privacy-preserving Nonlinear Data Similarity Evaluation

The same idea can be applied in the nonlinear case. The square area value can still be a metric of the nonlinear data models. The difference is that we need to map the metric to a higher dimensional form, and seek a higher dimensional solutions for computing θ and L. To address this mapping, the whole process should be considered with the kernel functions. Instead of using the Eq. (6), the equations with nonlinear form $\sum_{s\in S} \alpha_s y_s K(\mathbf{x}_s, \mathbf{t}) + b = 0$ for computing the boundary points should be listed. We take the nonlinear polynomial as an example, the new boundary computation should computed from the $\sum_{s\in S} \alpha_s y_s (\sum_{k_1+\ldots+k_n=p} \binom{p}{k_1,\ldots,k_n} \prod_{1\le i\le n'} x_i^{k_i} t_i^{k_i}) + b = 0$. Then, the dimension is changed from n to $n' = \binom{n+p-1}{n-1}$, and $w'_i = \sum_{s\in S} \alpha_s y_s \binom{p}{k_i}$. Similarly to the linear case, these arguments can still be applied into Eq. (6) to compute the boundary points for the mapped high dimensional data spaces,

$$\begin{cases} w'_1 u + w'_2\alpha + \ldots + w'_{n'}\alpha + b = \sum_{s\in S} \alpha_s y_s \binom{p}{k_1} u + \\ \sum_{s\in S} \alpha_s y_s [\sum_{k_1+\ldots+k_n=p} \binom{p}{k_2,\ldots,k'_n} \prod_{2\le i\le n'} w'^{k_i}_i \{\alpha, \beta\}^{k_i}_i] \\ \qquad\qquad + b = 0 \\ \ldots \\ w'_1 u + w'_2\beta + \ldots + w'_{n'}\beta + b = 0 \end{cases}$$

Similarly, the centroids of the higher dimensional space can be computed from the boundary points as $\mathbf{m}_A$ and $\mathbf{m}_B$. Combine with the decision function coefficients vectors, the new square area value of nonlinear data similarity evaluation can be expressed as follows,

$$T^2 = \frac{1}{4}[(K(\mathbf{m}_A, \mathbf{m}_A) + K(\mathbf{m}_B, \mathbf{m}_B) - 2K(\mathbf{m}_A, \mathbf{m}_B))^2 + L_0^4] \\ [(1 - \frac{K^2(\mathbf{w}_A, \mathbf{w}_B)}{K(\mathbf{w}_A, \mathbf{w}_A)K(\mathbf{w}_B, \mathbf{w}_B)}) + \sin\theta_0^2]$$

where all computation of kernel functions can be transformed as the inner products of the expanded polynomials. Following with the procedures of linear case, the terms $K(\mathbf{w}_B, \mathbf{w}_B)$ and $K(\mathbf{m}_B, \mathbf{m}_B)$ can be directly sent to Alice from Bob. The simplified function is similar as Eq. (8), and it is a polynomial function of x_1, x_2, where $x_1 = r_{am}K(\mathbf{m}_A, \mathbf{m}_B)$ and $x_2 = r_{aw}K(\mathbf{w}_A, \mathbf{w}_B) + r_b$ from Bob. Finally, by mapping the nonlinear data to higher dimensional space with the kernel functions, the similarity evaluation result T can still be obtained by Bob with privacy preservations.

VII. Privacy Analysis

As we assumed, the users in this system are honest-but-curious. All the users honestly perform the steps, but they are curious to deduce additional knowledge from what they obtained. Under such assumptions, two different level privacy objectives should be considered in the proposed schemes.

- *Level 1 objective*: To prevent the explicit privacy leakage, during the computation between the learners and testers, their private values should not be exposed.
- *Level 2 objective*: To prevent the implicit privacy leakage, after the computation process, the private value of the learners or testers should not be exposed.

A. Data Classification Level 1 Privacy

We consider each step in our proposed scheme to ensure the level 1 privacy requirement. On the one hand, from Bob's point of view, he hides all the input values into n different random polynomials g_i, and the random polynomial is different at each time. If Alice wants to get $\tilde{t}_i$, she has to collect at least $(pq+1)$ different values from one stable g_i. However, g_i are different at each time, and they are hidden with random numbers among all the M values. Since the oblivious transfer is implemented, it is impossible for Alice to find out the specific positions of g_i in a feasible time. So, Bob's private value $\tilde{\mathbf{t}}$ is protected. On the other hand, Alice hides her p-degree polynomial into a pq-degree polynomial. Although Bob can use the $(pq+1)$ terms to reconstruct the univariate polynomial $B(v)$. The coefficients of $d(\mathbf{G}(v))$ have been added up with a random polynomial $h(v)$. So, Bob cannot get the coefficients back from the interpolation function $B(v)$, and thus Alice's information is protected.

B. Data Classification Level 2 Privacy

The main privacy challenge of the level 2 privacy exists in the possible retrieval of the learned model. In what follows, we analyze the privacy prevention of model rebuild and data fitting attacks on the learned model.

1) Model Rebuild Attack Prevention in the Basic Scheme: On the Alice side, after the whole process of our scheme, since it is not feasible to deduce g_i, what she gets are just random numbers. Even Alice colludes with other learners, she cannot reveal anything from them due to the embedded randomness in the learning process. On Bob's side, he gets a random value $r_a d'(\tilde{\mathbf{t}})$ after the computation process. Since Bob has no idea about r_a in our proposed scheme, what he gets is just a randomized value with the same sign as the original one. No matter how many classification results Bob gets, he cannot find out the exact decision function of Alice. Therefore, both Alice and Bob cannot retrieve any information from what they get by the model rebuild attack, from which their privacy is preserved.

2) Data Fitting Attack Prevention in the Improved Scheme: Unfortunately, the simple positive random scaler r_a cannot prevent the data fitting attack. No matter the classification result is positive or negative, the outputs of classification will keep same data distribution. From the geometric perspective, the randomization only blurs the distance from the test data point to the division plane. Because there is no randomization difference for positive or negative results, both two sides data points of the division plane have the same statistical average distance as the non-randomized ones. Thus, the data fitting attack can successfully retrieve the learned model.

In order to prevent this issue, we present an improvement to the basic scheme, where r_a is modified for different classification results. We reverse the roles of Alice and Bob, and let Alice become the receiver to receive the randomized classification result, and let Bob be the sender to obliviously compute the polynomial from his test data $\mathbf{t}$. At the end of the computation, Alice will obtain the randomized classification value $r_a d(\mathbf{t})$. Then, according to different signs of this value, Alice resizes the value by multiplying it by a different number. For example, if the result is positive, it will be resized with a multiple from $[10^3, 10^5]$. Otherwise, it will multiply with a number from $[10^{-3}, 10^{-5}]$. Finally, Alice sends this resized value to Bob. As a consequence, the model parameters or test data information is still protected, but the classification results received by Bob is resized with different scale of random numbers. As a result, the output value distribution will be deviated, such that the data fitting attack can only retrieve a biased model and the privacy of original model is preserved.

C. Model Similarity Evaluation Privacy

Level 1 privacy can be achieved in model similarity evaluation scheme as the same as in above. Due to the privacy preservation of the secure computation protocols, both Alice and Bob's model information will not be revealed to each other during the evaluation process.

For Level 2 privacy, we consider the values Alice and Bob get after the process. In terms of linear scenario, Alice only gets $|\mathbf{m}_B|^2$ and $|\mathbf{w}_B|^2$. In the nonlinear case, she gets $K(\mathbf{m}_B, \mathbf{m}_B)$ and $K(\mathbf{w}_B, \mathbf{w}_B)$. Since these values are not separable, Alice learns nothing about Bob. From Bob's perspective, beside the final result, all the values he gets have been randomized, such that they cannot help him retrieve Alice's information. For the final evaluation, the area value T is not helpful for decision function's retrieval. Because the area value has factors of both the included angle and the centroid distance. Except the two learned models are exactly same, which is unlikely to happen, Bob cannot differentiate these two factors from the final result. Since all the values obtained by Bob or Alice are randomized or integrated, nothing can be deduced and thus the privacy is protected.

VIII. Performance Evaluation

In this section, three different experiments are designed to show the performance of our scheme in both simulation and real-world datasets. The effectiveness of privacy preservation, the correctness of protected classification, the efficiency and feasibility of computation are evaluated in the experiments. We apply the LIBSVM [41] as our basic SVM learning process. Both linear and nonlinear polynomial kernel SVMs are tested. The program is written in C++ programs and Python 2.7. It is implemented in CentOS 6.7 operating system with GCC version 4.4.7. The desktop has 4.00 GHz Intel(R) Core(TM) i7-4790K CPU and 32 GB memory. All the data have been scaled to $[-1, 1]$ and each dimension value of data is 8 Bytes.

A. Simulation-based Evaluation of Privacy Preservation

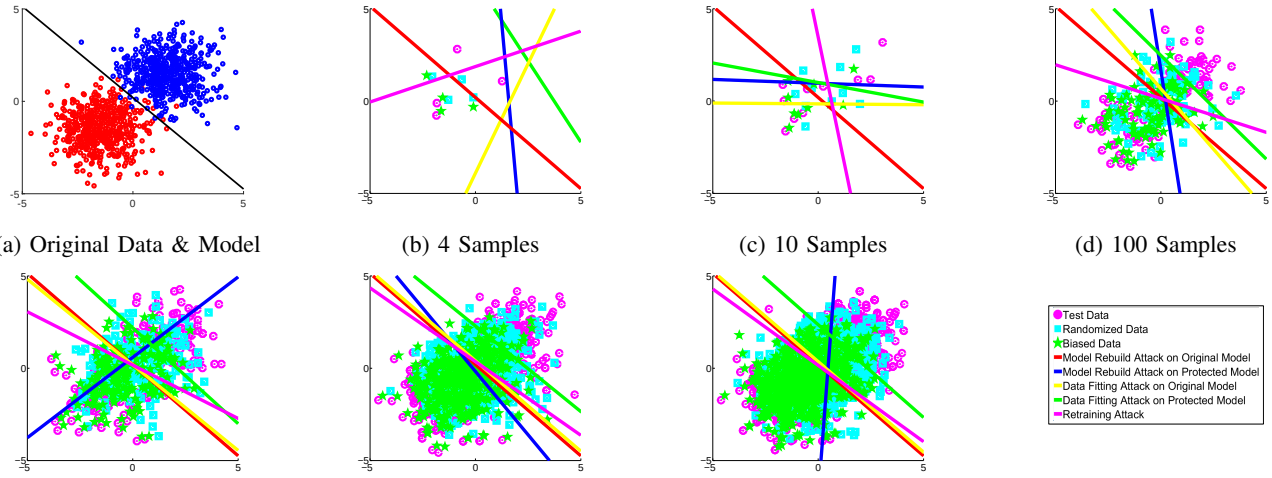
We first illustrate the effectiveness of privacy preservation in the proposed protection scheme. Specifically, we implement a simulation on the crafted data with different attack and protection schemes to demonstrate the Level 2 privacy preservation of the model. To have a better observation, we simulate the model by setting up a linear two-dimensional binary classifier with 1000 training samples. As shown in Fig. 7(a), these 1000 training samples are generated from two central points with the normal distribution. One central point $(-1.5, -1.5)$ is used to generated 500 negative samples, the other point $(1.5, 1.5)$ is used to generate the rest 500 positive points. The black line is the linear SVM decision function. Based on the outputs of this decision function, different attack approaches are applied for retrieving the original model. Fig. 7(b)-(g) show the effects of different attack approaches and corresponding protection schemes in different test data size. From the results, we can summarize the attack and protection performance as follow.

1) Without any protection, the model rebuild attack can accurately reconstruct the original model from few output values. As shown in Fig. 7(b), only 4 test data samples can rebuild the nearly same model as the original one.
2) After applying the protection of our basic scheme, the model rebuild attack retrieves the incorrect model parameters with its test data samples. The rebuilt models randomly stay in the data space for the incorrect classification.
3) With the increasing number of the test data samples, the reconstructions from the data fitting attack gradually close to the original model. From Fig. 7(b)-(c), the rebuilt model is disparate to the original model. However, with more information from the growing classification results, the rebuilt model become more similar to the original one.
4) After the protection of our improved scheme, the reconstructions from the data fitting attack are biased to the original model. As shown in Fig. 7(e)-(g), the rebuilt model is deviated to the correct classification model.
5) The retraining attack is also evaluated in the simulation. Compared to other attacks, our approach cannot prevent this attack. However, it requires more test data samples to rebuild the original model, which becomes more difficult to be deployed.

B. Experimental Results of Privacy-preserving Data Classification

In this experiment, we apply 17 different real-world datasets from [42] to evaluate the classification correctness, efficiency, and feasibility of our scheme. As a comparison, the Paillier additive homomorphic cryptosystem [22] is also implemented with a 128 bits key pair generator for evaluating the computational costs. In the nonlinear polynomial SVM, we set $a_0 = n^{-1}$, $b_0 = 0$, and $p = 3$. We set $c_0 = 1$ as default value for nonlinear Radial Basis Function (RBF) SVM.

1) Classification Accuracy: TABLE I shows the accuracy of original LIBSVM scheme on data classification from different datasets. At the same time, our scheme is implemented



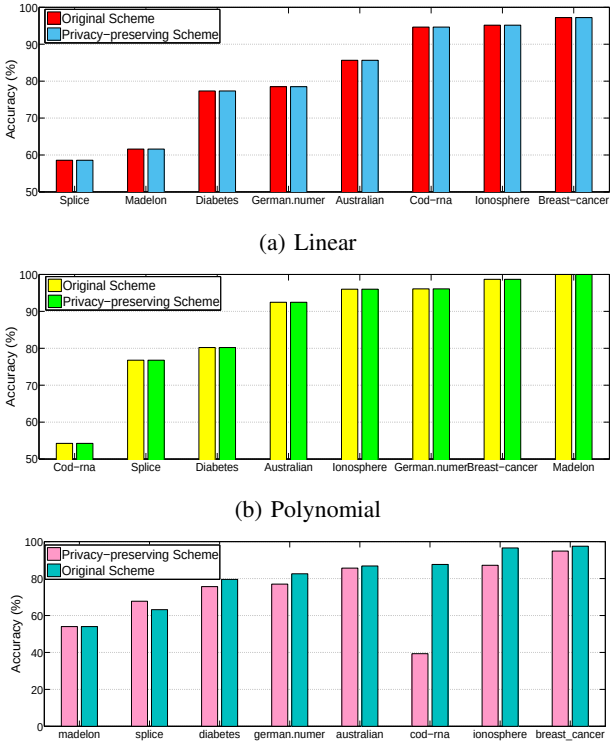
(a) Original Data & Model (b) 4 Samples (c) 10 Samples (d) 100 Samples

(e) 200 Samples (f) 500 Samples (g) 1000 Samples

Fig. 7: Model Attacks & Protection

TABLE I: Data Classification Accuracy

Dataset	Linear	Polynomial	RBF	Data Size	Data Space
splice	58.57%	76.78%	63.17%	2175	60
madelon	61.6%	100%	54%	2000	500
diabetes	77.34%	80.20%	79.56%	768	8
german .numer	78.5%	96.1%	82.6%	1000	24
a1a - a9a	82.51-84.69%	82.51-84.69%	84.23-85.08%	1605-32561	123
australian	85.65%	92.46%	86.81%	690	14
cod-rna	94.64%	54.25%	87.63%	59535	8
ionosphere	95.16%	96.01%	96.58%	351	34
breast-cancer	97.21%	98.68%	97.51%	683	10



(a) Linear

(b) Polynomial

(c) RBF ($p = 2$)

Fig. 8: Classification Accuracy

on these datasets to verify whether the classification correctness will be impacted by the additional process of privacy preservation in linear, polynomial, and RBF SVM. As shown in Fig. 8(a) and Fig. 8(b), compared with the original SVM, our proposed scheme predicts classes with almost the same accuracy for each dataset in both the linear and polynomial SVM. However, due to the approximation of Taylor expansion, our scheme has different classification accuracy than the original one in the RBF model. As shown in Fig. 8(c), we take the $p = 2$ for the approximation, which means we only take the first two terms of the Taylor expansions. In this figure, most approximated models keep the close accuracy as the original ones. The only exception is the dataset "cod-rna", which has a lower accuracy than the original model. This is because the "cod-rna" dataset has the most support vectors, i.e. about 19600. When we compute the Taylor expansion in Eq. (4), the value of cumulated support vectors will be far from the point $\mathbf{0}$. Therefore, only first two terms of the Taylor series cannot approximate the real value closely. To solve this issue, on the one hand, we can increase the value of p to take more terms from the expansion for approaching the real exponential value. On the other hand, we can decrease the scale of learning data. As shown in TABLE II, with the increase of p, the accuracy of the approximate classifier is improved. However, the required computation data space is also exponentially increased. TABLE III reflects the influence of decreasing the data range. The performance of the approximated model can be improved with smaller normalization range of data. But if the range is smaller than $[-0.1, 0.1]$, the accuracy of RBF classifier will have a sharp decline. As a result, we can slightly decrease the data range for improving the performance of approximated classifiers from Taylor expansion in the practical use.

2) Computational Cost: To analyze the efficiency and feasibility of the proposed scheme, we evaluate the computational

TABLE II: RBF Approximate Terms (Data Range $[-1, 1]$)

p	Normal SVM Acc.	PP SVM Acc.	Data Space
2	87.63%	39.34%	64
3	87.63%	68.95%	512
4	87.63%	75.73%	4096
5	87.63%	84.02%	32768

TABLE III: Data Range ($p = 2$) of RBF

Range	Normal SVM Acc.	PP SVM Acc.	Data Space
$[-1, 1]$	87.63%	39.34%	64
$[-0.5, 0.5]$	87.63%	87.64%	64
$[-0.2, 0.2]$	87.50%	87.50%	64
$[-0.1, 0.1]$	87.40%	87.40%	64
$[-0.08, 0.08]$	79.33%	79.33%	64
$[-0.06, 0.06]$	66.79%	66.79%	64

time cost in the datasets with different classification schemes.

TABLE IV: Computational Time Costs of Linear SVM (ms)

Dataset	Normal SVM	PP SVM	Paillier Cryptosystem
splice	90	1008	194553
madelon	1687	15897	1532880
diabetes	57	175	12556
german.numer	79	482	34040
a1a	198	3163	317741
a2a	356	4561	413906
a3a	573	6443	611233
a4a	1063	9437	968391
a5a	1737	12693	1195762
a6a	4726	22197	2212696
a7a	9424	32488	2916193
a8a	18819	45762	4697164
a9a	36825	64670	6517410
australian	53	225	15732
cod-rna	14660	43389	1079965
ionosphere	26	223	18269
breast-cancer	50	178	12881

First, we compare the time costs of the normal SVM classification, our approach, and the Paillier homomorphic cryptosystem on all the datasets. The results are shown in TABLE IV and they are evaluated from the linear SVM classifications, where the security parameters of our scheme are set as $q = 2$ and $k = 2$. From this table, our privacy-preserving scheme takes more time than the normal SVM classification process. This difference is an inevitable trade-off between the efficiency and the privacy preservation, which is mainly caused by the polynomial randomizations and oblivious transfers. However, compared to the results of Paillier cryptosystem, our approach prevents the huge computation loads from the complicated encryption, homomorphic addition, and decryption of model parameters and test data. To illustrate such difference, we plot the computational time costs of the datasets "a1a" to "a9a" in logarithm in Fig. 9, where they have same data feature dimensions but different test data volumes. From the figure, we can see that our approach requires much less time cost than the Paillier cryptosystem. Specifically, our approach takes about 65s to finish the privacy-preserving classification for 32MB data from dataset "a9a". However, the Paillier cryptosystem takes 6517s on the same dataset, which becomes infeasible to be practically applied in real applications.

Then, to evaluate the influence of security parameters in our scheme, we implement our privacy-preserving classification on the "a1a" dataset with different kernel functions. The security parameter q decides the degree of the randomization polynomials in our scheme, where larger q means that real information of the model will be hidden into the higher degree

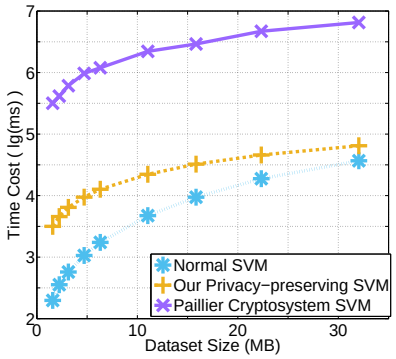
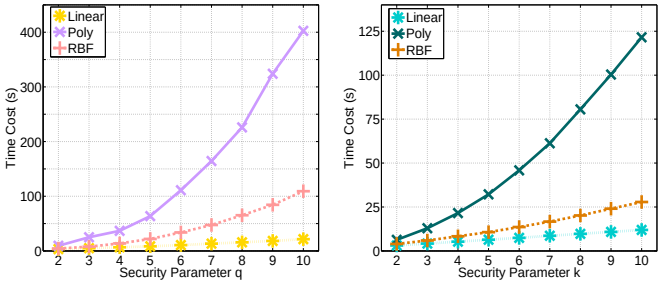


Fig. 9: Computational Cost Comparison of Different Schemes

polynomials. Meanwhile, since the test data is disguised by mk ($m = pq+1$) different values, the vector $\mathbf{G}(v)$ will become larger. Thus, with the growing of q, the time cost of our scheme will be increased. We keep $k = 2$ and evaluate the time costs of security parameter q, and the results are presented in Fig. 10(a). Due to the expansion terms of 3-degree polynomials, we can see that the change of q has a stronger influence on the computation of polynomial kernel. For example, from $q = 8$ to $q = 9$, it takes 22.6s to 32.4s for the polynomial kernel SVM, but it only needs 6.5s to 8.4s for the RBF kernel SVM. After that, we keep $q = 2$ and evaluate the influence of security parameter k in Fig. 10(b). Compared to q, the parameter k only participates in the generation of the vector $\mathbf{G}(v)$, which introduces fewer impacts on the computation.



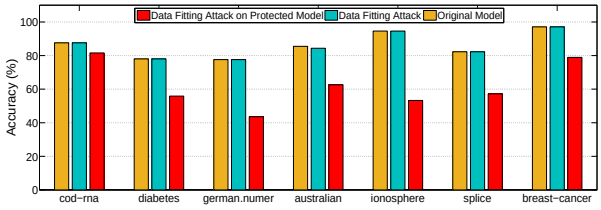
(a) q ($k = 2$) (b) k ($q = 2$)

Fig. 10: Computational Cost with Different Security Parameter

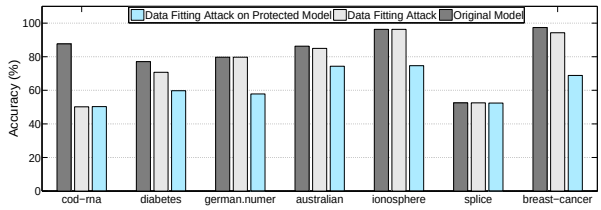
Although the increase of security parameters q and k requires more computational costs, it is not necessary to take large values of them to guarantee the privacy preservation. As analyzed in Section VII, the vector $\mathbf{G}(v)$ is randomly generated in each computation. The reconstruction of the test data cannot be achieved since it requires at least $(pq + 1)$ different values for each stable g_i. Also, it is infeasible to separate the $h(v)$ from $B(v)$ only from the $(pq + 1)$ terms of $A(v, \mathbf{G}(v))$. Therefore, as long as the condition $(pq + 1) > 1$ and $mk > m$ are satisfied, the privacy of data and model can be preserved. As a result, the selections of small values on q and k are sufficient to guarantee the privacy preservation.

3) Privacy Preservation: To illustrate the effectiveness of our privacy-preservation data classification scheme, we evaluate the performance of data fitting attack and corresponding protection schemes. The number of test samples we used is

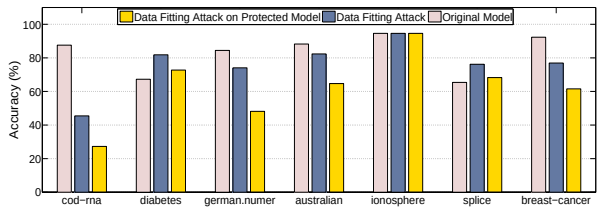
as twice as the data space dimension in the attack. As shown in Fig. 11, without the protection, most attacked model have the close accuracy of the original model. After implementing the privacy-preserving schemes, the accuracy of most attacks is decreased. To analyze the further effects, we also test the accuracy of the polynomial SVM on the "cod-rna" dataset with different test data samples. As shown in Fig. 12, with the growing number of test data, the accuracy of the data fitting attack is increased, but our protection scheme can still deviate the data fitting model and decrease its attack accuracy on the protected model.



(a) Linear



(b) Polynomial



(c) RBF ($p = 2$)

Fig. 11: Data Fitting Attacks & Protection

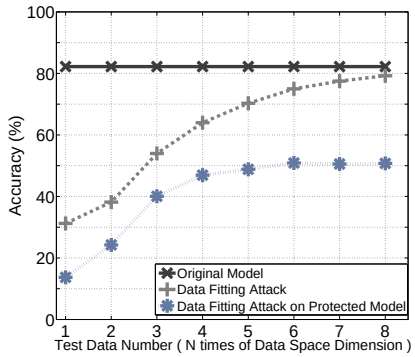


Fig. 12: Attack & Protection with Different Test Data

C. Experiments of Model Similarity Evaluation

To validate the effectiveness of our proposed model similarity evaluation approach, we implement the similarity computation in two different kinds of datasets.

1) Statistical Analysis: To evaluate the similarity, we need to compute the model with the same data feature dimensions. We split the dataset "diabetes" into 4 subsets as S_i, $1 \leq i \leq 4$ and train different models of each subset. Then, the Kolmogorov-Smirnov test [43] (KS test) is used as a statistical benchmark to support the correctness of our results for the similarity evaluation. KS test is used to see whether two data samples have the same distribution, where its evaluation result can be seen as the degree for describing the similarity from data statistical analysis. Since KS test can only work for two vectors, we test it on each data feature dimension for the split subsets. We get the average value over the 8 dimensions' KS test results as the final measurement. After that, we compute the cosine similarity and our approach between different model pairs to compare the similarity evaluation result. As shown in TABLE V, the scales of KS test results and our metric T are different, but they show the same trends of similarity evaluation in different pairs. However, the results of cosine similarity evaluation have a narrow range of output values and do not follow the value changes of KS test. Therefore, based on the statistical comparison, our similarity evaluation scheme provides a closer similarity measurement than the cosine similarity evaluation.

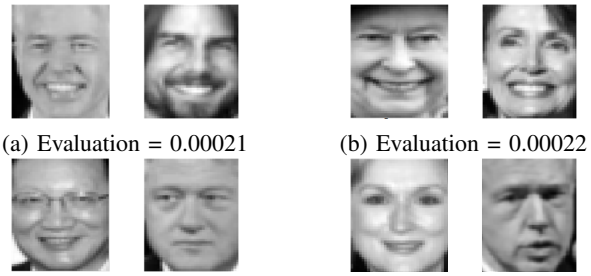
TABLE V: Privacy-preserving Data Similarity Evaluation

Subset Pair	KS Test Average	Cosine	Our Metric $T \times 10^3$
S_2 vs S_4	1.539	0.9552	5.858
S_3 vs S_4	2.757	0.9126	8.171
S_1 vs S_4	3.231	0.9417	9.470
S_2 vs S_3	6.264	0.9824	13.786
S_1 vs S_3	7.578	0.9814	27.736
S_1 vs S_2	8.557	0.9827	30.646

2) Empirical Analysis: As an empirical validation, we evaluate the similarity on the classification system of a real face image dataset, "Labeled Faces in the Wild" [44]. It includes about 19100 faces of 5749 different people with 250*250 pixels images stored in JPEG format. Before the evaluation, each image is tailored and resized into a 50×37 matrix. The classification model is built on the processed data in the RBF kernel SVM with the Eigenfaces [45] of Principal Component Analysis (PCA) transformation. The learned model includes multiple binary classifiers to construct a multi-class classification system with the one-to-all approach. Each person has its own binary classifier to predict whether the incoming data belongs to its own class.

Instead of directly comparing the images, our similarity evaluation compares the models that learned from the face images of the different person. We apply our similarity evaluation scheme to classifiers of 158 people in the dataset, where each of them has more than 10 face images. After the evaluation, the similarity results have a range of $[0.00021, 0.01673]$, and the example images of the most similar or different pairs are shown in Fig. 13. For the data of these images, it is difficult to assess the effectiveness of similarity evaluation statistically, but we can empirically validate the similarity from the present of figures. The similar pairs of face images have closer facial expression and feature, while the different pairs include the difference of expression, angle, and wear.

3) Computational Cost: Since the computation of privacy-preserving model similarity evaluation only calculates the parameters of decision functions, the cost actually equals



(a) Evaluation = 0.00021 (b) Evaluation = 0.00022

(c) Evaluation = 0.01645 (d) Evaluation = 0.01673

Fig. 13: Most Similar and Different Pairs

to a simplified one-time classification. Also, compared to larger requests for data classification, the requirement of model similarity evaluation is negligible. Therefore, instead of investigating the computational cost with different volumes of similarity comparison requests, we observe the evaluation of different data space dimensions. The results are shown in Fig. 14, which evaluate on the linear SVM of different datasets. With the increase of data dimensions, there is a significant increment for the required time cost of our approach. This is because one additional dimension requires more random polynomials to hide the information than a simple multiplication in the normal scheme. For example, one-time privacy-preserving similarity evaluation requires 0.26ms in 10-dimensional data space, while it takes 7.94ms in the 500-dimensional data space. Fortunately, in the real applications, the requests on the model similarity evaluation are usually limited and it will not require much time cost in total.

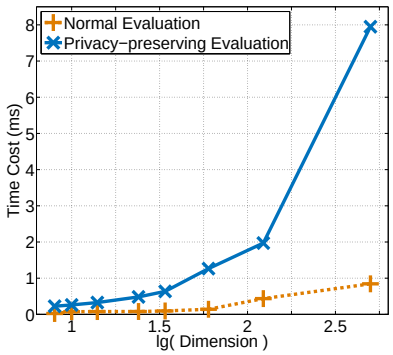


Fig. 14: Computational Cost of Similarity Evaluation

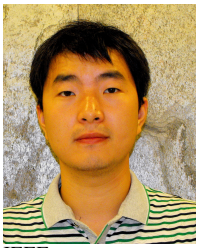
IX. Conclusion

In this paper, we propose an approach to preserve the model privacy for the distributed system in both data classification and model similarity evaluation. In the data classification scheme, by applying the oblivious evaluation of multivariate polynomial approach, the classification data and learned models are protected between the learning parties and testers. In the model similarity evaluation scheme, a novel metric is proposed for representing the closeness between different learned models with the privacy-preserving scheme. We also show the correctness, feasibility, and efficiency of our proposed scheme via extensive experiments.

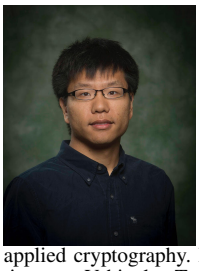
References

[1] N. Padhy, D. Mishra, R. Panigrahi *et al.*, "The survey of data mining applications and feature scope," *arXiv preprint arXiv:1211.5723*, 2012.
[2] E. Alpaydin, *Introduction to machine learning*. MIT press, 2014.
[3] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing." in *USENIX Security*, 2014, pp. 17–32.
[4] K. Chen and L. Liu, "Privacy preserving data classification with rotation perturbation," in *Data Mining, Fifth IEEE International Conference on*. IEEE, 2005, pp. 4–pp.
[5] O. L. Mangasarian and E. W. Wild, "Privacy-preserving classification of horizontally partitioned data via random kernels." in *DMIN*, 2008, pp. 473–479.
[6] B. Liu, Y. Jiang, F. Sha, and R. Govindan, "Cloud-enabled privacy-preserving collaborative learning for mobile sensing," in *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*. ACM, 2012, pp. 57–70.
[7] O. L. Mangasarian, "Privacy-preserving linear programming," *Optimization Letters*, vol. 5, no. 1, pp. 165–172, 2011.
[8] 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.
[9] C. C. Aggarwal and S. Y. Philip, *A general survey of privacy-preserving data mining models and algorithms*. Springer, 2008.
[10] C. Orlandi, A. Piva, and M. Barni, "Oblivious neural network computing via homomorphic encryption," *EURASIP Journal on Information Security*, vol. 2007, p. 18, 2007.
[11] J. Vaidya, H. Yu, and X. Jiang, "Privacy-preserving svm classification," *Knowledge and Information Systems*, vol. 14, no. 2, pp. 161–178, 2008.
[12] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *The Journal of Machine Learning Research*, vol. 11, pp. 1663–1707, 2010.
[13] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.
[14] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson, "Optimal parameter selection for the alternating direction method of multipliers (admm): quadratic problems," *Automatic Control, IEEE Transactions on*, vol. 60, no. 3, pp. 644–658, 2015.
[15] D. Lowd and C. Meek, "Adversarial learning," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 2005, pp. 641–647.
[16] 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.
[17] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *USENIX Security*, 2016.
[18] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. ACM, 2017, pp. 506–519.
[19] X. Wu, M. Fredrikson, S. Jha, and J. F. Naughton, "A methodology for formalizing model-inversion attacks," in *Computer Security Foundations Symposium (CSF), 2016 IEEE 29th*. IEEE, 2016, pp. 355–370.
[20] "Data leakage in healthcare machine learning," https://healthcare.ai/data-leakage-in-healthcare-machine-learning/, accessed: 2017-05-22.
[21] K.-P. Lin and M.-S. Chen, "Releasing the svm classifier with privacy-preservation," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 2008, pp. 899–904.
[22] Y. Rahulamathavan, R. C.-W. Phan, S. Veluru, K. Cumanan, and M. Rajarajan, "Privacy-preserving multi-class support vector machine for outsourcing the data classification in cloud," *Dependable and Secure Computing, IEEE Transactions on*, vol. 11, no. 5, pp. 467–479, 2014.
[23] M. Wilber, T. E. Boult *et al.*, "Secure remote matching with privacy: Scrambled support vector vaulted verification (s 2 v 3)," in *Applications of Computer Vision (WACV), 2012 IEEE Workshop on*. IEEE, 2012, pp. 169–176.
[24] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data." *IACR Cryptology ePrint Archive*, vol. 2014, p. 331, 2014.
[25] L. Xie, I. M. Baytas, K. Lin, and J. Zhou, "Privacy-preserving distributed multi-task learning with asynchronous updates," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 1195–1204.
[26] K. Lin, S. Wang, and J. Zhou, "Collaborative deep reinforcement learning," *arXiv preprint arXiv:1702.05796*, 2017.

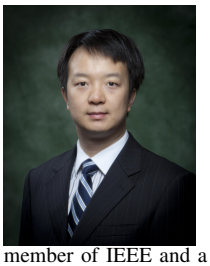
[27] S. K. Gupta, S. Rana, and S. Venkatesh, "Differentially private multi-task learning," in *Pacific-Asia Workshop on Intelligence and Security Informatics*. Springer, 2016, pp. 101–113.
[28] X. Jin, P. Luo, F. Zhuang, J. He, and Q. He, "Collaborating between local and global learning for distributed online multiple tasks," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 2015, pp. 113–122.
[29] S. Liu, S. J. Pan, and Q. Ho, "Distributed multi-task relationship learning," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 937–946.
[30] 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.
[31] K. Xu, Y. Guo, L. Guo, Y. Fang, and X. Li, "My privacy my decision: Control of photo sharing on online social networks," *IEEE Transactions on Dependable and Secure Computing*, no. 1, pp. 1–1.
[32] 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.
[33] M. Von Arb, M. Bader, M. Kuhn, and R. Wattenhofer, "Veneta: Serverless friend-of-friend detection in mobile social networking," in *Networking and Communications, 2008. WIMOB'08. IEEE International Conference on Wireless and Mobile Computing,*. IEEE, 2008, pp. 184–189.
[34] R. Lu, X. Lin, X. Liang, and X. Shen, "A secure handshake scheme with symptoms-matching for mhealthcare social network," *Mobile Networks and Applications*, vol. 16, no. 6, pp. 683–694, 2011.
[35] R. Zhang, J. Zhang, Y. Zhang, J. Sun, and G. Yan, "Privacy-preserving profile matching for proximity-based mobile social networking," *Selected Areas in Communications, IEEE Journal on*, vol. 31, no. 9, pp. 656–668, 2013.
[36] B. Scholkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.
[37] M. O. Rabin, "How to exchange secrets with oblivious transfer." *IACR Cryptology ePrint Archive*, vol. 2005, p. 187, 2005.
[38] M. Naor and B. Pinkas, "Efficient oblivious transfer protocols," in *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2001, pp. 448–457.
[39] C.-K. Chu and W.-G. Tzeng, "Efficient k-out-of-n oblivious transfer schemes with adaptive and non-adaptive queries," in *Public Key Cryptography-PKC 2005*. Springer, 2005, pp. 172–183.
[40] T. Tassa, A. Jarrous, and Y. Ben-Ya'akov, "Oblivious evaluation of multivariate polynomials," *Journal of Mathematical Cryptology*, vol. 7, no. 1, pp. 1–29, 2013.
[41] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.
[42] J. Platt *et al.*, "Fast training of support vector machines using sequential minimal optimization," *Advances in kernel methodssupport vector learning*, vol. 3, 1999.
[43] H. W. Lilliefors, "On the kolmogorov-smirnov test for normality with mean and variance unknown," *Journal of the American Statistical Association*, vol. 62, no. 318, pp. 399–402, 1967.
[44] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," University of Massachusetts, Amherst, Tech. Rep. 07-49, October 2007.
[45] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*. IEEE, 1991, pp. 586–591.



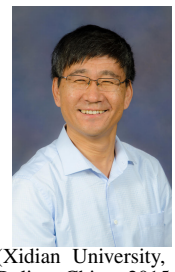
Qi Jia received his B.E. degree in Communication Engineering from Beihang University (Beijing University of Aeronautics and Astronautics) in 2014. He received the MS degree electrical and computer engineering from Binghamton University and is continuing his work towards the PhD. degree. His research 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 (M14) 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 is currently serving as the editor of IEEE Transactions on Vehicular Technology. 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.



Zhanpeng Jin (SM15) received the Ph.D. degree in electrical engineering from the University of Pittsburgh in 2010. He was a Postdoctoral Research Associate with the University of Illinois at Urbana-Champaign. He is currently an Associate Professor in the Department of Electrical and Computer Engineering at the Binghamton University, State University of New York (SUNY). His research interests include mobile and wearable computing in health, biometrics, neural engineering, neuromorphic computing, and low-power sensing. He is a senior member of IEEE and a member of 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).