2016 IEEE 36th International Conference on Distributed Computing Systems

# Privacy-preserving Data Classification and Similarity Evaluation for 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*—Data classification is a widely used data mining technique for big data analysis. By training massive data collected from the real world, data classification helps learners discover hidden data patterns. In addition to data training, given a trained model from collected data, a user can classify whether a new incoming data belongs to an existing class; or, multiple distributed entities may collaborate to test the similarity of their trained results. However, due to data locality and privacy concerns, it is infeasible for large-scale distributed systems to share each individual's datasets with each other for data similarity check. On the one hand, the trained 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 a privacy-preserving data classification and similarity evaluation scheme for distributed systems. With our scheme, neither new arriving data nor trained models are directly revealed during the classification and similarity evaluation procedures. The proposed scheme can be applied to many fields using data classification and evaluation. Based on extensive real-world experiments, we have also evaluated the privacy preservation, feasibility, and efficiency of the proposed scheme.**

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

## I. Introduction

Tremendous amount 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 people dig out useful information from massive messages [1]. For example, by analyzing customers' behaviors, supermarket owners can discover the association among different items and reflect customers' shopping habits, such as Bestmart [2]. By training patients' health records, hospitals can build disease classification models to diagnose or prognosticate new diseases. As an important analytical method, in all kinds of data mining approaches, machine learning based data classification plays a significant role. Generally speaking, data classification mainly consists of two parts, data training and testing. During the training process, the training entity (trainer) applies learning methods to learn the data structure and divides the data into various classes according to different data features. In the testing phase, a subset of data not used in training process will be used to test for the data classification generality, i.e., to test the accuracy of new data classification. After training and testing are done, trained model can be used to classify the future data to the corresponding classes.

To better articulate our intuition behind the proposed scheme, we consider a practical application scenario that highly relies on the success of data classification for distributed systems. With the growth of online shopping, data classification techniques have been widely adopted in electronic commerce (e-commerce). Every e-commerce company (trainer) has huge amount of online sale records containing customers' information. From the records, data classification techniques can be applied to figure out the relationship between the quantity and features of sold items. Taking clothing sale as an example, companies can utilize its online sale records to identify the sale trend. By applying it, clothes sellers (clients) can test whether their design follows the popular trend, which help sellers stay competitive with the sale fashions by following the direction of customer requirements. Besides, since the sale trending model is unique for each company from its own sale records, it can also be used to evaluate the market similarity across different companies. When a company wants to find a business partner, it can firstly compare its sale trending model with others' to have a basic understanding of their market similarities. Then, if two companies have similar sale trends, they may cooperate with each other to start new business activities.

Unfortunately, stumbling block is the privacy issue that apparently exists in the above data classification and similarity evaluation process. On the one hand, since the sale records are only stored and possessed by each company, no matter how the sale trending model is used, it should be kept undisclosed to other entities. On the other hand, the sellers' designs are also private and should be protected properly. As a result, it is necessary to preserve the privacy of both the trained models and clients' data in data classification and similarity evaluation processes. However, most works [3]–[13] focus on preserving the privacy in the data training process, where the training data are protected, but the trained models are published to everyone. Only a few works pay attention to the privacy issues of trained model [14]–[17]. However, their schemes are either inefficient or lack of practical applications. In this paper, we propose a practical and efficient privacy-preserving distributed data classification and similarity evaluation scheme to address the aforementioned issues. We mainly rely on the oblivious evaluation of multivariate polynomials approach and the oblivious transfer protocol between the trained model and the new arriving data to hide the private information by using different random polynomials.

In summary, we have made the following major contribu-

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 CNS-1423165.

1063-6927/16 $31.00 © 2016 IEEE
DOI 10.1109/ICDCS.2016.94

tions:

- Our proposed privacy-preserving data classification scheme ensures that the trainers can conduct data classification successfully without exposing their trained models to the clients, while the clients keep their data in private.
- We propose privacy-preserving geometric metric to assess the closeness of different trained models. The proposed scheme can privately compute data similarities between different trainers without exposing their trained models to each other.
- As a basic data classification approach, Support Vector Machine (SVM) is used to analyze the main challenges in both linear and nonlinear data distribution scenarios.
- Two different privacy levels are analyzed in our scheme for both data classification and similarity evaluation.

The rest of this paper is organized as follows. The related works are briefly reviewed in Section II. In Section III, we provide preliminaries for better understanding. Then, our proposed schemes are explained in Section IV and Section V for different scenarios. The privacy analysis and experimental evaluation are provided in Section VI. Finally, we conclude this paper in Section VII.

## II. Related Work

### A. Privacy Preservation in Data Classification

Privacy issues in data classification can generally categorized into training and classification (testing) privacy. The issues raised in training process are about how to protect the training data and privately compute the trained models. Randomization approaches are applied in the early time. In [3]–[6], they mix the training data by using random rotation perturbation or random matrix. The training process is manipulated on the randomized data and the correct trained model can be acquired after the de-randomization. Other approaches are based on the cryptographic methods. Several systems [7]–[10] propose solutions by solving the training process on the encrypted training data. The homomorphic cryptosystem is applied so that the optimization problems can still be solved from 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 [5], which is inefficient and impractical. Recently, most research works move further steps to the distributed data systems. For fully distributed systems in [11]–[13], the Alternating Direction Method of Multiplier (ADMM) are combined with the SVMs to achieve private data training among individual users. ADMM is one solution to augmented lagrangian multiplier method which achieve the separable iterative optimization problem. This property can be used on the SVM for private training process and each individual user only trains with their own data. Xu *et. al.* [18]–[20] analyze the distributed privacy preserving SVMs in both the horizontally and vertically partitioned data by modifying the ADMM schemes. The assumption behind all these papers is that the training data are private but the trained model can be public.

However, the trained model should also be protected as a reflection of the data structures. To solve this privacy problem, Lin *et. al.* [14] reorganizes the expansion of RFB kernel function, and a polynomial format is expressed as an approximate value. It puts all the support vectors into one number such that each support vector cannot be retrieved. However, the real polynomial coefficients are still revealed to the receivers. Rahulamathavan *et. al.* [15] propose a scheme where the Paillier cryptosystem is applied to transform the SVM decision function into an encrypted form. The classification sample is also encrypted. All the computations are operated on the ciphertext and only the clients who holds the private key can decrypt the classified result. It achieves the protection of both trained models and classification samples. Unfortunately this cryptographic approach introduces too much complexity for the computations and it is not practical to be used in the real application. The most related work to our approach is [17]. 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 protected the privacy of trained model and clients inputs. However, the multiple cryptosystems would introduce additional procedures such as different key management. Compared to their approach, we apply a uniform OMPE method without additional procedures for the classifications and we also provide the similarity evaluation to different trained models.

### B. Privacy Preservation in Data Similarity Evaluation

Our privacy-preserving similarity evaluation is close to matching systems. Privacy-preserving matching system 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 [21], [22], they propose coarse-grained privacy-preserving matching systems. In their descriptions, the matching attribute are derived from a big public set so that the matching problems become to the Private Set Intersection problems. In [23], Zhang *et. al.* propose a fine-grained matching system, where all attributes are graded in different values. The fine-grained matching process can get the matching result with the detail value comparison. However, the grading process is not described clearly in this paper. Users can input any values they want for the matching purpose, and this subjective inputs can lead to a totally mismatched result. In our proposed solutions, the similarity evaluation is based on the trained models, which are the reflections of inherent behaviors, and it is 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 finding out a binary classifier based on the training data [24]. Here, we assume data $\mathbf{x} \in \mathbb{R}^n$ is an $n$ dimensional vector, and each dimension $x_i, 1 \leq i \leq n$, is one feature of the data. For each training data $\mathbf{x}$, it has a label $y \in \{+1, -1\}$ indicating which class it belongs to. The training process is to find a hyperplane that divides the data space with the largest margin distance between different data label groups.

*1) Linear Classification:* For linear classification problem, it is possible to find a linear hyperplane divides the data into two classes in training process. Then, 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 $\mathbf{w}^T\mathbf{t} + b < 0$). The trained decision function $d(\mathbf{t})$ can be summarized as

$$d(\mathbf{t}) = \mathbf{w}^T\mathbf{t} + b$$

where $\mathbf{w}$ and $b$ are acquired by solving the optimization problem on maximizing margin distance, and they indicate the division direction and position of the hyperplane. Normally, the decision function can be reformulated 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 $S$ is the set of support vectors. Specifically, support vectors $\mathbf{x}_i, 1 \le i \le |S|$ are the vectors located at the margin of the data classes, and $y_i, 1 \le i \le |S|$ are the corresponding labels of the support vectors. Here, $\alpha_i, 1 \le i \le |S|$ are the Lagrangian multiplier parameters. A class label can be assigned to the sample $\mathbf{t}$ by computing the classification function, where the **sign**$(\cdot)$ is the sign function,

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

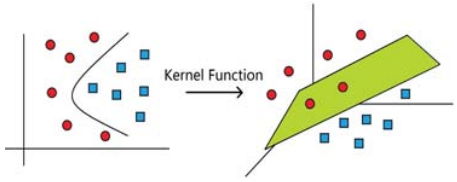



Fig. 1: Kernel Method

*2) Nonlinear Classification:* In the nonlinear classification, as shown in Fig. 1, the data is distributed nonlinearly and it can hardly find a linear hyperplane to divide the data perfectly in original dimension. To solve this problem, kernel methods are applied [24]. Generally, kernel functions can map the low dimensional data to a higher dimensional space. In the higher dimensional space, the data become more separated or better structured, in which the linear SVM method can be applied. For the nonlinear SVM decision function, the dot product $\mathbf{x}^T\mathbf{t}$ term is substituted by kernel functions as a mapped higher dimensional dot product. 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 [25]–[27], receiver and sender, for transmitting messages. By executing this protocol, the receiver only gets one part of information from the sender without knowing other parts, while the sender does not know which part is known by the receiver. The oblivious transfer 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 $\sigma$ and wishes to get the $\sigma$th 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 $\sigma$.
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 [28]. 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 secure 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 $P(\boldsymbol{\alpha}) = Q(0, \boldsymbol{\alpha}) = Q(0, \mathbf{S}(0)) = R(0)$.

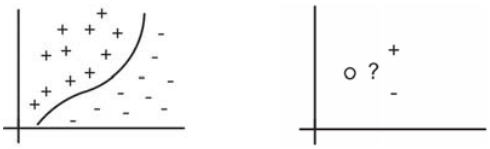
### D. Threat Model

To illustrate the privacy challenges in our scheme, we put forward two assumptions for the threat model. On the one hand, 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. On the other hand, we assume the collusion may happen among trainers or clients. A trainer/client may collude

with other trainers/clients to speculate more messages on one client/trainer. However, the trainers should not collude with the clients. Besides, we suppose clients are not powerful enough to manipulate the training process, which means they cannot retrain an imitative model from their classified data.

## IV. Privacy Preserving Data Classification

In this section, we propose a privacy-preserving data classification scheme for computing the classes of the clients' samples. Assuming Alice is the trainer who holds the training dataset and has derived a classifier, while Bob is the client who wants to figure out which class his data sample belongs to. As shown in Fig. 2, the black curve in (a) can be reviewed as the trained model of Alice, it divides the data space into two different classes. The circle in Fig. 2 (b) is representing 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 trained 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) Trained Pattern (b) Testing Data

Fig. 2: Data Classification

### A. Privacy-preserving Linear Data Classification

In linear data classification problem, the data is located linearly in the data space. Alice has trained 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.

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

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 is holding the trained model $d(\mathbf{t})$ as an $n$ variates polynomial function with degree 1. Bob initiates the request with Alice for privately computing the classification result. Then, Alice generates a random univariate polynomial, $h(u)$, with 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 classification 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 needs to make 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 above 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 $M$ nonzero 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 [28]. 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,

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

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 to Alice, while Alice still 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 changed 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, kernel function is used to map the lower dimensional vectors to a higher dimension. For different data distribution, various kernel functions can be applied to train a more accurate model. 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: $K(\mathbf{x}, \mathbf{y}) = e^{||\mathbf{x}-\mathbf{y}||^2}$
- Sigmoid Kernel: $K(\mathbf{x}, \mathbf{y}) = \mathbf{tanh}(\mathbf{x}^T \mathbf{y} + c_0)$

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:

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

- 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 large number $p$ to approximate the infinity. 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. Taking the nonlinear polynomial kernel function as an example, with $a_0 = 1$, $b_0 = 0$, and $c_0 = 0$, the data classification processes should be updated to the nonlinear scenario and the corresponding decision function is,

$$\begin{aligned} & d(\mathbf{t}) = \sum_{s\in S} \alpha_s y_s (\mathbf{x}^T \mathbf{t})^p + b \\ & = \sum_{s\in S} \alpha_s y_s (x_1 t_1 + ... + x_n t_n)^p + b \\ & = \sum_{s\in S} \alpha_s y_s ( \sum_{k_1+...+k_n=p} \binom{p}{k_1, ..., k_n} \prod_{1\leq i\leq n} x_i^{k_i} t_i^{k_i}) + b = \\ & \sum_{k_1+...+k_n=p} \{ [\sum_{s\in S} (\alpha_s y_s \binom{p}{k_1, ..., k_n} \prod_{1\leq i\leq n} x_{si}^{k_i})] [\prod_{1\leq i\leq n} t_i^{k_i}] \} + b \end{aligned}$$

In this case, $d(\mathbf{t})$ is an $n' = \binom{n+p-1}{n-1}$ variates polynomial, and input terms $\prod_{1\leq i\leq n} t_i^{k_i}$ are the combination multiplications of original variates $t_i$. If we treat each term as a variate $\tau_j$, $\tau_j = \prod_{1\leq i\leq n} t_i^{k_i}, 1 \leq i \leq n, 1 \leq j \leq n'$, we can rewrite $d(\mathbf{t})$ as $d(\boldsymbol{\tau})$, where $d(\boldsymbol{\tau})$ is just an $n'$ variates $p$-degree polynomial. Similarly, we can apply the OMPE approach to this transformed polynomial to achieve the secure computation. In the trained 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 $g_i(v)$ are 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,

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

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

## V. Privacy Preserving Data Similarity Evaluation

As we mentioned earlier, data similarity evaluation is to calculate the similarities among different trainers. As shown in the Fig. 3, since trainers have different training data, their trained models could be different from each other. In order to compare two different models, we propose a privacy-preserving data similarity evaluation scheme. To evaluate the similarity, the first step is to find a metric that can properly represent the closeness of different trained models. In our scheme, we combine both factors of the direction and position of trained models as a metric. Then, we explain how to

compute this metric with the privacy preservation of different trainers. Similarly, our scheme will be elaborated in both linear and nonlinear scenarios.

(a) Trained Pattern 1 (b) Trained Pattern 2

Fig. 3: Data Similarity Evaluation

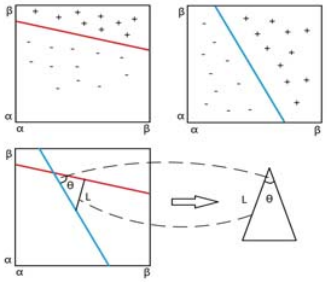


Fig. 4: Bounded Data Space and Isosceles Triangle Construction

### A. Similarity Metric

*1) Cosine Similarity:* Considering the geometry concepts in 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. By doing so, the 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 and the hyperplane is unbounded, 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 coefficients vector of the high dimensional decision function, the cosine value, or called cosine similarity, can be calculated as follows,

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

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 $n$ dimensional data space. Two hyperplanes may have different angles and locations in the data space. As shown in Fig. 4, 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. Apparently, only comparing their directions cannot truly reflect the real closeness. For this reason, the decision function of trained 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 for two high dimensional bounded hyperplanes instead of two unbounded ones.

*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 $\theta$ and the centroid distance is $L$. By combining these two elements, as shown in Fig. 4, we can construct an isosceles triangle with vertex angle equals to $\theta$ and legs equals 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 relative to the sine value of $\theta$ 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 a 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 constant values as angle $\theta_0 << 90^\circ$ and distance $L_0$ on the original terms. These two values can be public, and the square area value becomes to follows,

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

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 trained 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 Data Similarity Evaluation

To privately compute this metric triangle area, we continue to use Alice and Bob as an example to compare their trained 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. (5), it should treat one dimension value as a variable at each time, and the other dimensions should be arranged as different combinations of

$\alpha$ and $\beta$. 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

$$\begin{cases} w_1u + w_2\alpha + ... + w_n\alpha + b = 0 \\ w_1u + w_2\alpha + ... + w_n\beta + b = 0 \\ ... \\ w_1u + w_2\beta + ... + w_n\alpha + b = 0 \\ ... \\ w_1u + w_2\beta + ... + w_n\beta + b = 0 \end{cases} \tag{5}$$

All the feasible $u$ in these equations, where $\alpha \leq u \leq \beta$, can make up the first dimension boundary points with the corresponding $\alpha$ and $\beta$. 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 $\gamma$, $\delta$ 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. (4) between Alice and Bob can be written as follows,

$$\begin{aligned} T^2 &= \frac{1}{4}(L^4 + L_0^4)(\sin^2\theta + \sin^2\theta_0) \\ &= \frac{1}{4}[(|\mathbf{m}_A|^2 + |\mathbf{m}_B|^2 - 2\mathbf{m}_A \cdot \mathbf{m}_B)^2 + L_0^4] \\ &\quad [(1 - \frac{(\mathbf{w}_A \cdot \mathbf{w}_B)^2}{|\mathbf{w}_A|^2 \times |\mathbf{w}_B|^2}) + \sin\theta_0^2] \end{aligned} \tag{6}$$

*2) Area Computation:* As so far, the privacy-preserving similarity evaluation problem has been transformed to 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 expand the Eq. (6) as a multivariate polynomial, we take some steps to simplify the computations.

To get rid off the fraction part, Bob can send $|\mathbf{m}_B|^2$ and $|\mathbf{w}_B|^2$ to Alice directly. Since these two terms are the vector module squares, Alice cannot reveal any specific 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. Actually, 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 $c_1 = ||\mathbf{m}_A||^2 + ||\mathbf{m}_B||^2$, $c_2 = L_0^4$, $c_3 = (||\mathbf{w}_A||^2 \times ||\mathbf{w}_B||^2)^{-1}$, $c_4 = 1 + \sin\theta_0^2$, 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 as a two variates polynomial function of $x_1$ and $x_2$ as follows,

$$T^2(x_1, x_2) = \frac{1}{4}[(c_1 - 2d_1x_1)^2 + c_2][c_4 - c_3d_2(d_3 + x_2)^2] \tag{7}$$

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

We apply the same idea in the nonlinear data distributed scenario. 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 $\theta$ and $L$. To address this mapping, the whole process should be considered with the kernel functions. Instead of using the Eq. (5), 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. The new square area value of nonlinear data similarity evaluation can be expressed as follows,

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

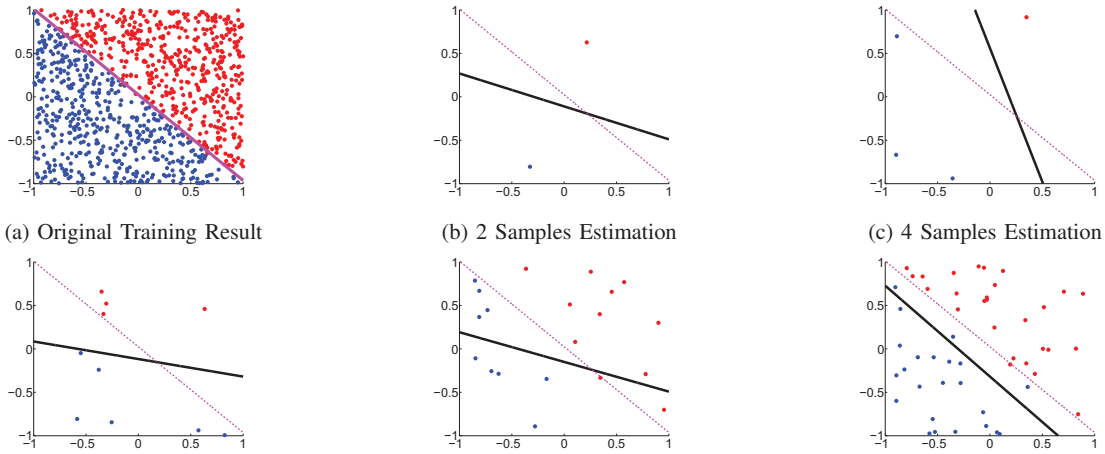
where the terms $K(\mathbf{w}_B, \mathbf{w}_B)$ and $K(\mathbf{m}_B, \mathbf{m}_B)$ can be directly sent to Alice from Bob. The other steps are basically similar to the linear process, but all the dot products should be replaced by the kernel functions. The simplified function is similar as Eq. (7), and it is a polynomial function of $x_1, x_2$. However, these two values can be obtained by computing $x_1 = r_{am}K(\mathbf{m}_A, \mathbf{m}_B)$ and $x_2 = r_{aw}K(\mathbf{w}_A, \mathbf{w}_B) + r_b$ from Bob, and other constants are also updated. 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.

## VI. Performance Evaluation

### A. Privacy Analysis

As we assumed, the users in this system are honest-but-curious. All the users honestly perform the steps in our scheme. But they are curious and try to deduce additional knowledge from what they obtained. Also, one user may collude with other users during the process. Under such assumptions, there are two different level privacy objectives will be considered in our proposed schemes.

- *Level 1 objective*: During the computation between the trainers and clients, their private values should not be exposed to each other in each step.



(a) Original Training Result (b) 2 Samples Estimation (c) 4 Samples Estimation

(d) 10 Samples Estimation (e) 20 Samples Estimation (f) 50 Samples Estimation

Fig. 5: Model Estimation

- *Level 2 objective*: After the computation process, even other participants are colluded, the private value of the trainers or clients should not be exposed.

*1) Data Classification Privacy:* For Level 1 privacy, each step in our proposed scheme needs to be considered. On the one hand, from Bob's point of view, he hides all the input values into $n$ different random polynomials $g_i$, and each time the random polynomial is different. 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 in 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 to the random polynomial $h(v)$. So, Bob cannot get the coefficients back from the interpolation function $B(v)$ and Alice's information is protected.

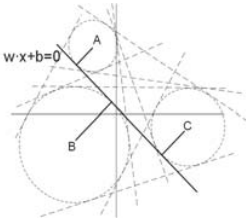



Fig. 6: Decision Function Retrieval

For the Level 2 privacy issue, our proposed system can also prove the information of Alice and Bob is secure. On the Alice side, after the whole process, since it is not feasible to deduce $g_i$, what she gets are just random numbers. Even Alice colludes with other trainers, she cannot reveal anything from them due to the embedded randomness in the training process. On Bob's side, he gets a random value $r_a d'(\tilde{\mathbf{t}})$ after the computation process. If this the random value $r_a$ is not applied, Bob can get the real distance from his points to the decision plane. In this way, it is easy to retrieve the $n$ dimensional linear classifier $d(\mathbf{t})$ by just collecting $(n+1)$ different distance values. Geometrically, as shown in Fig. 6, we can draw a circle at each data point with radius equals to the distance. Between each two circles, there are four tangents can be found. So, in the linear two dimensional data space, by finding the common tangent line of distance circles, only three points are enough for the reconstruction of the linear decision function. However, since Bob has no idea about $r_a$ in our proposed scheme, what he gets is just a randomized value with same sign as the original one. No matter how many other clients Bob colludes with, he cannot find out the exact decision function of Alice. Therefore, theoretically, both Alice and Bob cannot retrieve any information from what they get, from which their privacy of them are preserved.

To illustrate above issue, we simulate a model by setting up a linear two dimensional binary classifier as the decision function of Alice with 1000 training samples as the training inputs. Then, Bob colludes with other users and he directly uses randomized classification results to estimate the decision function. As shown in Fig. 5, the line in figure (a) is Alice's decision function based on all the training data. The solid line in other figures, Fig. 5 (b)-(f), represents an estimation for decision function based on different classified results. We can see that the estimations are different from the original one, and the solid lines are randomly lying on different directions and positions in the data space. No matter how many classification results obtained by Bob, the estimation is still irregular and keeps rambling.

*2) Similarity Evaluation Privacy:* Level 1 privacy in data similarity evaluation scheme can be achieved with the same analysis in the data classification. Both Alice and Bob's classifier information will not be revealed to each other during the evaluation process.

For Level 2 privacy, we consider what values Alice and Bob get after the process. In terms of nonlinear scenario, Alice only gets $K(\mathbf{m}_B, \mathbf{m}_B)$ and $K(\mathbf{w}_B, \mathbf{w}_B)$. Since these two values are not separable, Alice learnings nothing about Bob. From Bob's

perspective, beside the final result, all the values he gets have been randomized, so that they cannot help him retrieve Alice's information. For the final evaluation, the area value $T$, it is not helpful for decision function's retrieval. Because the area value has factors of both the included angle sine value and the centroid distance. Except the two trained 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, there is no valuable information can be deduced and the privacy is protected.

### B. Experiment Result

In the experiments, we apply the LIBSVM [29] as our basic SVM training process. Both linear and nonlinear polynomial kernel SVMs are tested in our scheme. The program is written in C++ programs and 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.

*1) Data Classification:* We apply 17 different datasets from [30] to verify our scheme. In the nonlinear polynomial SVM, we set $a_0 = n^{-1}$, $b_0 = 0$, and $p = 3$ as default values.

TABLE I: Data Classification Accuracy

| Dataset | Linear | Polynomial | Testing Size | Dimensions |
|---|---|---|---|---|
| splice | 58.57% | 76.78% | 2175 | 60 |
| madelon | 61.6% | 100% | 2000 | 500 |
| diabetes | 77.34% | 80.20% | 768 | 8 |
| german.numer | 78.5% | 96.1% | 1000 | 24 |
| a1a - a9a | 82.51-84.69% | 82.51-84.69% | 1605-32561 | 123 |
| australian | 85.65% | 92.46% | 690 | 14 |
| cod-rna | 94.64% | 54.25% | 59535 | 8 |
| ionosphere | 95.16% | 96.01% | 351 | 34 |
| breast-cancer | 97.21% | 98.68% | 683 | 10 |

TABLE I shows all the accuracy of the original LIBSVM scheme on the data classification. The data size and feature dimensions are different between datasets. For each dataset, we analyze the performance of both linear and nonlinear SVM. Then, our scheme are implemented on these datasets to illustrate the functionality. As shown in Fig. 7 and Fig. 8, comparing with the original SVM, our proposed scheme predicts the classes with same accuracy for each dataset. It proves that our scheme can guarantee the same prediction functionality as the original SVM scheme.

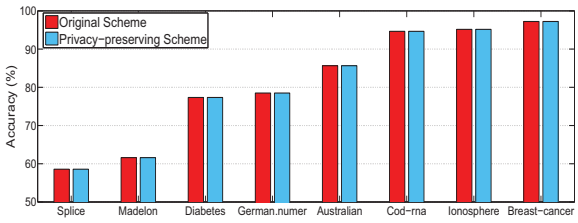



Fig. 7: Accuracy of Linear Data Classification

To analyze the efficiency of our proposed scheme, we take the datasets "a1a" to "a9a" as samples. Since they have same feature dimensions and an increment of data size, we can assess their time cost. Fig. 9 shows the time cost for the original SVM and our proposed privacy-preserving approach on both linear and nonlinear schemes. The horizontal axis is the classification data size and the vertical axis is the time cost for different SVM schemes. The relations between linear and nonlinear SVMs are similar in original scheme and our approach. Since we add the random polynomial to the process, our schemes take more time than the original scheme. According to the experiment result, it is about 4 times more than the original schemes. We can further reduce the time cost by generating random polynomials before the scheme.

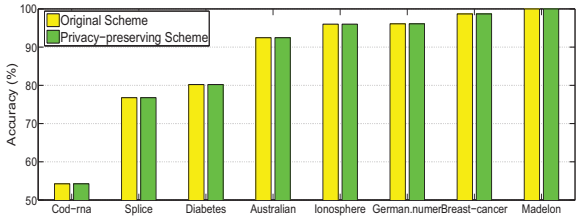



Fig. 8: Accuracy of Nonlinear Data Classification

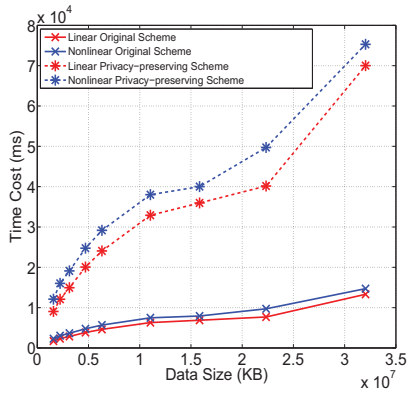



Fig. 9: Computational Cost Comparison of Classification

*2) Similarity Evaluation:* In order to show the correctness of our similarity evaluation, we split 4 subsets from the dataset "diabetes" as $S_i$, $1 \leq i \leq 4$ and each subset has 192 items of data. The Two-sample Kolmogorov-Smirnov test [31] method is used as a comparison of similarity measurement for our scheme. Kolmogorov-Smirnov test is a statistic test method to see whether two data samples have the same distribution. It has a test output result as the degree for describing the similarity. Since this test can only work for two vectors, we test it on each data feature dimension for the split subsets. Then, we get the average value over the 8 dimensions' K-S test results as the measurement. We also evaluate the similarity of different subset pairs by our scheme. Because the data boundary is $[-1, +1]$, the area value of metric triangle $T$ will be too small. So, we amplify the value with $10^3$ as the final result. As shown in TABLE II, the scale of K-S test results and our similarity $T$ are different, but they show the same trend of comparisons between the subsets, which means our similarity evaluation scheme can provide a correct measurement on the dataset.

The privacy-preserving similarity evaluation takes more

TABLE II: Privacy-preserving Data Similarity Evaluation

| Subset Pair | K-S Test Average | Our Scheme $10^3T$ |
|---|---|---|
| $S_1$ vs $S_2$ | 8.557 | 30.646 |
| $S_1$ vs $S_3$ | 7.578 | 27.736 |
| $S_1$ vs $S_4$ | 3.231 | 9.470 |
| $S_2$ vs $S_3$ | 6.264 | 13.786 |
| $S_2$ vs $S_4$ | 1.539 | 5.858 |
| $S_3$ vs $S_4$ | 2.757 | 8.171 |

computation cost than the ordinary evaluation. Fig. 10 compares one evaluation cost of the ordinary and privacy-preserving similarity evaluation schemes. It also shows the time cost growth with the increment of data space dimensions. Since one additional dimension requires more random polynomials to hide the information than a simple multiplication in the ordinary scheme, we can find that the changing of dimensions has more impact on the privacy-preserving evaluations in terms of the computation time.

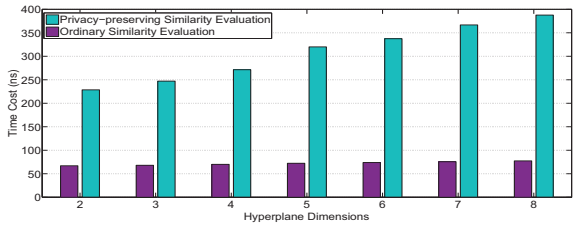



Fig. 10: Computational Cost Comparison of Similarity Evaluation

## VII. Conclusion

In this paper, we propose privacy-preserving schemes for data classification and data similarity evaluation. In the data classification scheme, by applying the oblivious evaluation of multivariate polynomial approach, the classification data and trained models are protected between the training parties and clients. In the data similarity evaluation scheme, a novel metric is proposed for representing the closeness between different trained models with 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] IBM, "Big data in retail industry - examples in action: Bestmart," 2013. [Online]. Available: http://lab.sowre.com/2013/06/big-data-in-retail-examples-in-action-bestmart-slides

[3] 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.

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

[5] 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.

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

[7] 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.

[8] C. C. Aggarwal and S. Y. Philip, *A general survey of privacy-preserving data mining models and algorithms*. Springer, 2008.

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

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] B. Scholkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[25] M. O. Rabin, "How to exchange secrets with oblivious transfer." *IACR Cryptology ePrint Archive*, vol. 2005, p. 187, 2005.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] J. Platt *et al.*, "Fast training of support vector machines using sequential minimal optimization," *Advances in kernel methodssupport vector learning*, vol. 3, 1999.

[31] 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.