Privacy-Preserving SVM using Nonlinear Kernels on Horizontally Partitioned Data *

Hwanjo Yu
Department of Computer Science
University of Iowa, Iowa City
hwanjoyu@cs.uiowa.edu

Xiaoqian Jiang
Department of Computer Science
University of Iowa, Iowa City
xjia@cs.uiowa.edu

Jaideep Vaidya
Management Science & Information Systems
Rutgers University, Newark
jsvaidya@rbs.rutgers.edu

ABSTRACT

Traditional Data Mining and Knowledge Discovery algorithms assume free access to data, either at a centralized location or in federated form. Increasingly, privacy and security concerns restrict this access, thus derailing data mining projects. What we need is distributed knowledge discovery that is sensitive to this problem. The key is to obtain valid results, while providing guarantees on the non-disclosure of data. Support vector machine classification is one of the most widely used classification methodologies in data mining and machine learning. It is based on solid theoretical foundations and has wide practical application. This paper proposes a privacy-preserving solution for support vector machine (SVM) classification, PP-SVM for short. Our solution constructs the global SVM classification model from the data distributed at multiple parties, without disclosing the data of each party to others. We assume that data is horizontally partitioned – each party collects the same features of information for different data objects. We quantify the security and efficiency of the proposed method, and highlight future challenges.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*; H.2.7 [**Database Management**]: Database Administration—*Security, integrity, and protection*; H.2.4 [**Database Management**]: Systems—*Distributed databases*

General Terms

Security,Algorithms

Keywords

privacy-preserving data mining, support vector machine

1. INTRODUCTION

*Portions of this work were supported by a Rutgers Research Resources Committee grant

SAC'06 April 23-27, 2006, Dijon, France

Data Mining has many applications in the real world. One of the most important and widely found problem is that of classification. For example, fraud detection can be posed as a classification problem. Specifically, take the case of identifying fraudulent credit card transactions. Banks collect transactional information for credit card customers. Due to the growing threat of identity theft, credit card loss, etc. identifying fraudulent transactions can lead to annual savings of billions of dollars. Deciding whether a particular transactions is true or false is a classification problem. Many such problems abound in various diverse domains.

Currently, classifiers run locally or over data collected at a single central location (i.e., on a data warehouse). The accuracy of a classifier usually improves with more training data. Data collected from different sites is especially useful, since it provides a better estimation of the population than the data collected at a single site. However, privacy and security concerns restrict the free sharing of data. There are both legal and commercial reasons to not share data. For e.g., HIPAA laws [13] require that medical data not be released without appropriate anonymization. Similar constraints arise in many applications; European Community legal restrictions apply to disclosure of any individual data [8]. In commercial terms, data is often a valuable business asset. For example, complete manufacturing processes are trade secrets (although individual techniques may be commonly known). Thus, it is increasingly important to enable privacy-preserving distributed mining of information. However, central accumulation of summaries or obfuscated models might be considered reasonable as long as the original data is not revealed.

Support Vector Machine (SVM) is one of the most actively developed classification methodology in data mining and machine learning. It provides salient properties such as the margin maximization and nonlinear classification via kernel tricks and proven to be effective in many real-world applications [32, 4, 5].

We assume the data is *horizontally partitioned* [16, 36]: each party holds different data objects having the same features. Numerous practical problems fall under this data model. For instance, different banks collect data for their customers. The features collected, such as age, gender, balance, average monthly deposit, etc. are the same for all banks.

We propose a privacy-preserving SVM classification solution, PP-SVM for short, which constructs the global SVM classification model from the data distributed at multiple parties. The data of each party is kept private, while the final model is constructed at an independent site. This independent site then performs classification of new instances. This makes a lot of sense in the horizontally partitioned context – Consider a clearing house for the prior consortium of banks. The clearing house is an independent entity, unrelated

to any of the banks. The classification model is constucted at the clearing house while preserving the privacy of the individual data from each of the banks. When a bank has a new instance it wants to classify, it goes through a secure protocol with the clearing house to classify just this instance. The clearing house learns nothing. This would allow all of the banks to leverage the global data without compromising on privacy at all.

A PP-SVM solution for *linear kernels* has been proposed in [36]. However, it is not extendible to nonlinear kernels because its solution is based on the optimization formulation of the proximal SVM [10]. Our approach is new and specifically designed for SVM *nonlinear kernels*. Our method is currently limited to the nonlinear kernels whose kernel matrix can be constructed from the gram matrix. However, most popularly used nonlinear kernels such as polynomial or RBR kernels can be computed from the gram matrix. Our method generates the same SVM classification model as when the data is centralized. We quantify the security and efficiency of our algorithm.

We also assume that each party does not collude and does follow the proposed protocol correctly. While the assumption of non-collusion is quite strict, it enables an efficient algorithm. We later discuss this issue in more detail.

Our method runs on data represented by binary feature vectors such that a data object $x = (e_1, ..., e_n)$ and e_i = 1 or 0. Binary feature vector has been a popular representation in many learning methodologies (e.g., Rule learning like k-DNF and CNF, decision tree, SVM) and used in a wide domains of applications (e.g., automatic diagnosis [18], classifying Web pages and emails [35], mining graphs [34]). In any case, it is possible to transform a dataset in any representation to a binary feature vector representation. For our experiments, we use the SPECT dataset from the UCI repository [1]. The preprocessed binary feature representation of the dataset is used. Indeed, this has shown a better performance in classification than the original continuous feature representation [1].

The kernel matrix is the central structure in a SVM. It contains all the necessary information for the learning algorithm and fuses information about the data and the kernel. *As such, the kernel matrix plays a role as the intermediate profile that does not disclose any information on local data but can generate the global model.* To construct the global SVM model without disclosing the data, our method securely computes the *kernel matrix* from the distributed data; The global SVM model can then be generated from the kernel matrix. We assume that knowledge of the kernel matrix is necessary to efficiently perform classification. Later, we do discuss exactly what information is leaked by the kernel matrix itself, and ways to circumvent this restriction.

The kernel matrix for polynomial or RBF kernels can be computed from the *gram matrix* which is composed of the *dot products* of every data pair. Thus, we first securely compute the gram matrix to construct the kernel matrix. The key challenge here is the issue of scalability: To build the gram matrix, we need to compute the dot products of every data *pair*. Thus, the communication cost is key. The paper is organized as follows. We first overview SVM (Section 2) and present our method (Section 3). We then empirically show the practicality of the overall approach in Section 4. Related work is discussed in Section 5. We conclude our study and present directions for future research in Section 6.

2. SVM OVERVIEW

Notation

All vectors are column vectors unless transposed to a row vector by a prime superscript $'$. The scalar (dot) product of two vectors x and y in a n-dimensional space is denoted by $x'y$ or $x \cdot y$ and the 2-norm of x is denoted by $||x||$. An $m \times n$ matrix $\mathcal{A}$ represents m data points in a n-dimensional input space. An $m \times m$ diagonal matrix $\mathcal{D}$ contains the corresponding labels (*i.e.*, +1 or -1) of the data points in $\mathcal{A}$. (A class label $\mathcal{D}_{ii}$, or d_i for short, corresponds to the i-th data point x_i in $\mathcal{A}$.) A column vector of ones of arbitrary dimension is denoted by e. The identity matrix of arbitrary dimension is denoted by $\mathcal{I}$.

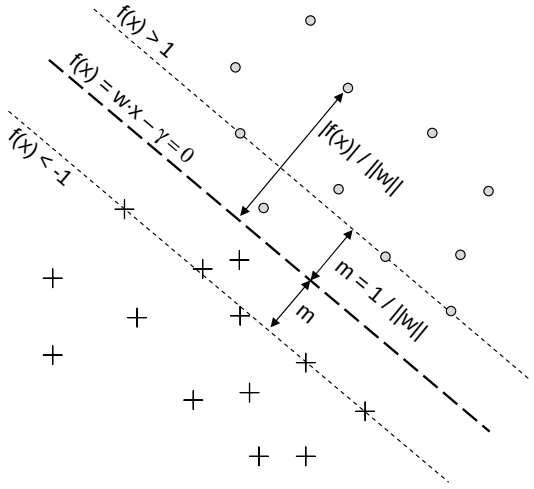

Figure 1: The separating hyperplane that maximizes the margin. ('o' is a positive data point, *i.e.*$f(\text{`}o') > 0$, and '+' is a negative data point, *i.e.*$f(\text{`}+') < 0$.)

Overview

To first overview the standard SVM classification, consider a linear binary classification task, as depicted in Figure 1. For this problem, SVM finds the separating hyperplane ($w \cdot x = 0$) that maximizes the *margin*, denoting the distance between the hyperplane and closest data points (*i.e.*, support vectors).

The margin is denoted by $\frac{1}{||w||}$ as illustrated in Figure 1. To maximize the margin while minimizing the error, the standard SVM solution is formulated into the following primal program [32, 10]:

$$\min_{w,y} \quad \tfrac{1}{2}w'w + \nu e'y \tag{1}$$

$$s.t. \quad \mathcal{D}(\mathcal{A}w - e\gamma) + y \geq e \quad and \;\; y \geq 0 \tag{2}$$

which minimizes the reciprocal of the margin (*i.e.*, $w'w$) and the error (*i.e.*, $e'y$). The slack variable y is larger than zero when the point is on the wrong side or within the margin area. The soft margin parameter ν is tuned to balance the margin size and the error. The weight vector w and the bias γ will be computed by this optimization problem. The class of a new data x will be determined by $f(x) = w'x - \gamma$, where the class is *positive* if $f(x) > 0$, otherwise *negative*.

Applying the Lagrange multipliers on the primal problem, we have the following dual problem [32]:

$$\min_{\alpha} \quad \tfrac{1}{2}\alpha'\mathcal{Q}\alpha - e'\alpha \tag{3}$$

$$s.t. \quad 0 \leq \alpha_i \leq \nu \;\; and \;\; \textstyle\sum_i d_i\alpha_i = 0, \quad i = 0, ..., m \tag{4}$$

where $d_i (i.e., \mathcal{D}_{ii})$ and α_i are respectively the given class label and the coefficient for data vector x_i. The coefficients $\alpha =<$

$\alpha_1, ..., \alpha_m >$ are to be computed from this dual problem. An $m \times m$ matrix $\mathcal{Q}$ is computed by the dot product of every data pair such that $\mathcal{Q}_{ij} = K(x_i, x_j)d_i d_j$ where $K(x_i, x_j) = x_i \cdot x_j$ for linear SVM.

Once α is learned, γ can be computed from Eq.(2) and $w = \sum \alpha_i d_i x_i$. The support vectors are the data vectors $\{x_i\}$ such that the corresponding coefficients $\alpha_i > 0$. The classification function $f(x) = w'x - \gamma$ becomes $\sum \alpha_i d_i x_i \cdot x - \gamma$ for linear SVM. For nonlinear SVMs, $f(x) = \sum \alpha_i d_i K(x_i, x) - \gamma$, where we can apply a nonlinear kernel for $K(x_i, x)$, e.g., $K(x_i, x) = exp(-\frac{||x_i - x||^2}{g})$ for RBF kernel, $K(x_i, x) = (1 + x_i \cdot x)^p$ for polynomial kernel.

3. PRIVACY-PRESERVING SVM CLASSIFICATION

In our environment, the m data points in the matrix $\mathcal{A}$ are assumed to be partitioned and distributed over multiple parties; the corresponding class labels in $\mathcal{D}$ are also distributed likewise.

To build the global SVM model over the distributed data without disclosing the data of each party to the others, we *securely compute the kernel matrix $\mathcal{K}$ where $K_{ij} = K(x_i, x_j)$*: First, to build an SVM model, we need to solve the dual problem in Eq.(3) and (4), which requires the $m \times m$ matrix $\mathcal{Q} = K(x_i, x_j)d_i d_j$ to solve it. Sharing the class labels $\mathcal{D}$ would reveal the size of each class in each party. However, it does not reveal any information on the data itself as long as *the kernel matrix $\mathcal{K} = K(x_i, x_j)$ is securely computed.* Thus, we assume that the class labels are shared, and we focus on the secure computation of the kernel matrix.

To securely compute the kernel matrix, *we compute the gram matrix $\mathcal{G}$ securely where $G_{ij} = x_i \cdot x_j$*; The nonlinear kernel matrices like polynomial or RBF can be computed from the gram matrix $\mathcal{G}$, as the data vectors appears only as dot products in both polynomial and RBF kernels: Polynomial kernel = $(x_i \cdot x_j + 1)^p$, RBF kernel = $exp(-\frac{||x_i - x_j||^2}{g}) = exp(-\frac{x_i \cdot x_i - 2x_i \cdot x_j + x_j \cdot x_j}{g})$. Thus, the gram matrix $\mathcal{G}$ is enough to compute the kernel matrix $\mathcal{K}$ for polynomial and RBF kernels.

To securely compute the global gram matrix $\mathcal{G}$, we must securely compute the dot product over every data pair. This is possible using a secure dot product computation method. However, most existing secure dot product computation methods [6, 27, 14, 24] are inefficient or insecure to be applied for computing the gram matrix. The one exception is the method developed by Goethals et al [12], which is fully secure and is actually suitable for this work. (Section 5 discusses this issue in detail.) For now, we introduce an alternate efficient and secure way to compute the gram matrix $\mathcal{G}$.

The key idea is to use a *secure set intersection cardinality* [30] to securely compute the dot products: To use the secure set intersection cardinality, we revise the representation of a binary feature vector into an ordered set such that the elements of the set are the indexes of '1' in the original vector. For example, suppose vector $x_1 = (1, 0, 1, 1, 0, 0, 0, 1, 0, 1)$ and $x_2 = (0, 0, 0, 1, 0, 0, 0, 1, 1, 0)$ in a 10-dimensional space. Then, they will be represented as ordered sets $x_1' = \{1, 3, 4, 8, 10\}$ and $x_2' = \{4, 8, 9\}$ respectively. *Now the dot product of two vectors becomes equivalent to the size of the set intersection between the two sets.* That is, $x_1 \cdot x_2 = |x_1' \cap x_2'| = 2$. On this revised representation of data, we securely compute the set intersection cardinality using commutative one-way hash function. Computing the set intersection cardinality using commutative one-way hash functions has been proven to be secure [30].

There is also the challenge of scalability when multiple parties (e.g., more than two parties) are involved: Unlike computing the intersection of all parties presented in [30], we need to perform the intersection between *all pairs* of the parties, because the gram matrix contains the dot product of every *data pair*. With k participating parties, the communication cost becomes $O({}_kC_2 \approx k^2)$. This is one of the reasons for avoiding standard dot product computation.

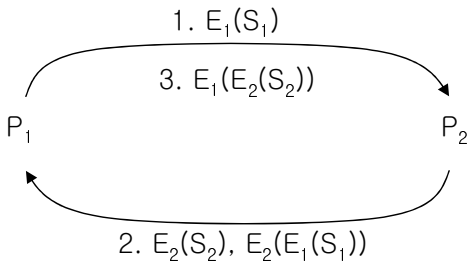

Figure 2: A protocol to exchange the datasets securely between two parties encrypted by the keys of both parties.

In Section 3.2, we present a scalable and secure protocol that computes the kernel matrix in linear time, without disclosing any data. We first discuss computing the set intersection cardinality securely over *two parties* (Section 3.1).

3.1 Securely Computing the Size of Set Intersection over Two Parties

Assume two parties P_1 and P_2 hold sets S_1 and S_2 respectively. The problem is to securely compute the *size* of the intersection set, $|S_1 \cap S_2|$. Computing the set intersection cardinality over two parties using commutative one-way hash function has been proven to be secure [30].

The key idea behind the method is simple: Parties P_1 and P_2 encrypt their sets with their own private keys E_1 and E_2 respectively using the commutative one-way hash function, exchange them, and encrypt them again with the other party's key. Both parties P_1 and P_2 will obtain $E_2(E_1(S_1))$ and $E_1(E_2(S_2))$. Due to the commutative property of the one-way hash function, $E_2(E_1(x_1)) = E_1(E_2(x_2))$ only when $x_1 = x_2$, where x_1 and x_2 are elements in S_1 and S_2 respectively. To be more specific, $\exists i, j; E_2(E_1(x_1 i)) = E_1(E_2(x_2 j)) only when x_1 i = x_2 j$, where $x_1 i$ and $x_2 j$ are features in vector x_1 and x_2 respectively. Thus, it is possible to compute the number of equivalent elements without decrypting the sets.

It is possible for each party to have multiple sets since each set represents one data vector. For convenience of explanation, however, we assume each party holds only one set. It is straightforward to extend it for when each party hold multiple sets.

The formal definition of such a hash function can be found in [30, 22].

DEFINITION 1. *A commutative keyed one way hash function (CKHF) is a one way hash function h_k, parameterized by a secret key k, with the following additional property:*

1. *commutative hashing - given two instances of a keyed hash function h_k parameterized with 2 different keys k_1 and k_2 and an input x, $h_{k_1}(h_{k_2}(x)) = h_{k_2}(h_{k_1}(x))$.*

A commutative public key encryption scheme such as Pohlig-Hellman can be used to generate a hash function satisfying all our requirements. The hash function h_{k_i} is simply encryption with E_i.

Figure 2 illustrates the protocol to securely exchange the sets between P_1 and P_2 encrypted by the keys of both parties. The following is the description of the protocol.

1. P_1 encrypts its set with its private key (= $E_1(S_1)$) and sends it to P_2.

2. P_2 encrypts the received set and its own set with its private key (= $E_2(E_1(S_1))$ and $E_2(S_2)$ respectively), saves a copy of the second set (= $E_2(E_1(S_1))$), and sends both sets to P_1 in that order.

3. P_1 saves the second set (= $E_2(E_1(S_1))$) and encrypts the first set with its private key (= $E_1(E_2(S_2))$), saves a copy of it, and sends it to P_2.

Once both parties P_1 and P_2 obtain $E_2(E_1(S_1))$ and $E_1(E_2(S_2))$, they can compute the set intersection cardinality between them without decryption due to the commutative property of the one-way hash function. The proof of the security of this set intersection cardinality method is presented in [30]. The astute reader will notice that this method is not fully secure – the number of 1s in a vector is revealed. This is a tradeoff made for efficiency. In general, this is not a big problem. However, where even this leakage is unacceptable, a completely secure method such as the one by Goethals et al.[12] should be used.

3.2 Secure and Scalable Protocol for Multiple Parties

Suppose we have k number of participating parties, $P_1, ..., P_k$, where P_1 is actually the independent entity which contributes no data to the process. P_1 is also the "initiator" of the protocol. The other parties activate only on receiving a message from another party. Each passive party P_i ($2 \leq i \leq k$) has a dataset S_i. As we discussed in the introduction, the initiator is the third party that participates in the protocol to receive the final model without contributing to training data.

Unlike computing the intersection of all parties presented in [30] (i.e., $|\cap S_i|$), we need to compute the intersection between all *pair* of parties except the initiator (i.e., $|S_i \cap S_j|$ for $2 \leq i \leq k$ and $2 \leq j \leq k$), in order to construct the global gram matrix; The gram matrix contains the dot product of every data pair.

A direct extension from the two-party protocol requires $O(k^2)$ times of communications, because it has to be done for every pair of parties. In this section, we present a scalable and secure protocol that scales linearly (= $O(k)$) for communication complexity.

The key idea is that the initiator (1) first collects the datasets encrypted by all the parties' keys, (2) computes the *kernel matrix* through the gram matrix using the set intersection cardinality, and Since computation is only done over encrypted data, data privacy is still protected. Note that the gram matrix actually reveals linear equations (if one of the parties data is known). However, since the initiator does not contribute to the training data, it cannot obtain any linear equations. The initiator only sees the dot products of unknown variables and thus nonlinear equations where the number of variables is always larger than the number of equations. Without colluding with one of the parties, the initiator cannot learn anything.

For convenience of notation,

- we specify $E_{i,j}(S)$ for $E_i(E_j(S))$.
- $E_{i\sim j}(S)$ will denote either $E_i(E_{i+1}(...(E_{j-1}(E_j(S)))))$ if $i < j$, or $E_i(E_{i-1}(...(E_{j+1}(E_j(S)))))$ if $i > j$. For instance, $E_{k-3\sim 1,k,k-1}(S)$ denotes $E_{k-3}(E_{k-2}(...(E_1(E_k(E_{k-1}(S))))...))$
- We mark * if S is "fully encrypted", i.e., encrypted by every key from P_2 to P_k regardless of the encryption sequences.

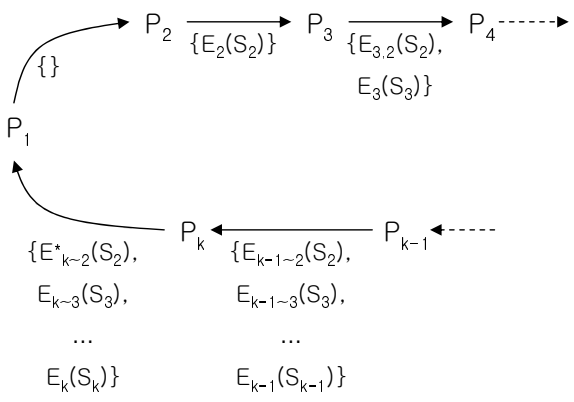

Figure 3: First Round of the Protocol.

For instance, $E^*_{2\sim k}(S) = E^*_{k\sim 2}(S)$ because of the commutative property of the one-way hash function.

From the first two rounds of communications, the initiator P_1 collects the fully encrypted datasets from all the other parties. Then, P_1 computes the gram matrix using the set intersection cardinality. This is done directly over the encrypted datasets. $E^*_{...}(S_i) = E^*_{...}(S_j)$ only when $S_i = S_j$ regardless of the encryption sequence due to the commutative property of the one-way hash function. P_1 then computes the kernel matrix from the gram matrix. Finally, P_1 now computes the global SVM model. This computation does not require either the original data vectors or the gram matrix. Neither the kernel matrix nor the gram matrix is ever sent to any of the parties. Thus, they never learn anything.

We now completely describe each round of our protocol. The full description of the protocol for the initiator P_1 and all other passive parties P_i is given in Table 1 and 2 respectively.

- **ROUND 1** (illustrated in Figure 3): The first round begins by the initiator P_1 sending an empty set to its neighbor P_2. Whenever a party P_i receives a set SS from P_{i-1}, it adds its own dataset S_i to SS and encrypts SS with its own key E_i, and send the revised $SS' = E_i(\{SS, S_i\})$ to P_{i+1} (or to P_1 if $i = k$.) At the end of the first round, P_1 will receive from P_k a set $SS = \{E^*_{k\sim 2}(S_2), E_{k\sim 3}(S_3), ..., E_k(S_k)\}$ such that S_i encrypted by E_k to E_i. Note that only the first element in SS is fully encrypted.

  During the first round, no dataset is disclosed, as the initiator and each passive party receive a dataset encrypted by at least one key unknown to them. Note that, to prevent P_1 from contributing to training set, P_2 must reject P_1's start request if P_1 sends a non-empty set in the first round.

- **ROUND 2** (illustrated in Figure 4): P_1 removes the first element $E^*_{k\sim 2}(S_2)$ from the SS and sends the rest to P_2. Once a party P_i receives the SS from P_{i-1}, it encrypts SS with E_i, and sends the first element $E^*_{i\sim 2,k\sim i+1}(S_{i+1})$ to P_1 and the rest to P_{i+1}. Note that the first element is now fully encrypted. P_1 will receive the fully encrypted S_{i+1} from P_i where $(2 \leq i \leq k-1)$. At the end of the second round, P_1 will obtain every dataset fully encrypted.

  During the second round, no dataset is disclosed, as each party again holds a dataset encrypted by at least one key unknown to it. The initiator is also unable to glean any information from a dataset since they are all fully encrypted.

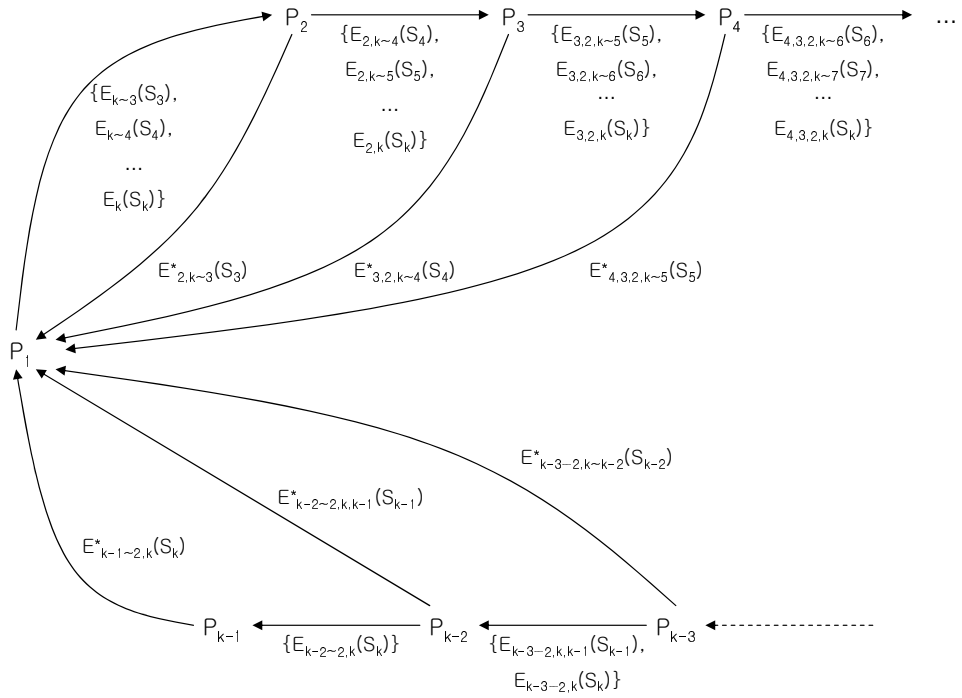

Figure 4: Second Round of the Protocol.

- **ROUND 3:** P_1 now computes the gram matrix from the datasets by performing the set intersection cardinality. It then computes the kernel matrix from the gram matrix, and finally computes the global SVM model from the kernel matrix.

Assuming that the participating parties do not collude with the initiator and do follow the given protocol, the initiator successfully acquires the global SVM model without disclosing any information on the private data of any party.

One could ask why the kernel matrix is not sent to the participating parties. The key problem is the fact that if the kernel is known, then any party could easily infer the gram matrix from the kernel matrix. In fact, with a *linear kernel*, the kernel matrix is the same as the gram matrix. Since the gram matrix is composed of dot products, and each dot product inherently reveals one equation in the feature set to a participant, each party could then induce linear equations from the kernel matrix. Since the number of data points will typically be much larger than the number of features, the equations can be solved to easily reveal the original data of the other parties.

Yu and Vaidya have [36] proposed a secure method for SVM *linear kernels* by utilizing the optimization formulation of the proximal SVM [10]. Our method complements it for SVM *nonlinear kernels*.

3.3 Secure Testing/Classification of the SVM Model

This section discusses how to securely perform testing / classification using the SVM model constructed by our protocol. An SVM model is represented by the support vectors, their coefficients α and the bias γ. After obtaining the kernel matrix and the class labels, as we discussed in Section 2, the initiator can solve the dual problem in Eq.(3) and (4), without the individual data vectors, in order to learn the coefficients α. γ can be computed from α. The support vectors are the vectors whose coefficients are greater than zero.

Initiator P_1

Round 1:

1. Send an empty set to P_2.
2. Wait to receive a new set SS from P_k. (SS will be $\{E^*_{k\sim 2}(S_2), E_{k\sim 3}(S_3), ..., E_k(S_k)\}$. Note S_2 is fully encrypted.)

Round 2:

1. Take out the first element $E^*_{k\sim 2}(S_2)$ from the SS and send the rest to P_2.
2. Wait to receive the fully encrypted S_{i+1} from P_i where $2 \leq i \leq k-1$. (Note that, at the end of Round 2, P_1 will acquire all the datasets $\{E^*_{...}(S_2), ..., E^*_{...}(S_k)\}$ fully encrypted.)

Round 3:

1. Construct the gram matrix by intersecting the encrypted sets to compute the required dot products.
2. Construct the kernel matrix from the gram matrix.
3. Construct the global SVM model from the kernel matrix.

Table 1: The protocol for the initiator P_1.

Party P_i**,** $2 \leq i \leq k$

Round 1:

1. Receive a set SS from P_{i-1}.
2. Add S_i to SS and encrypt SS with E_i such that the revised set $SS' = E_i(\{SS, S_i\})$.
3. Send SS' to P_{i+1} (or to P_1 if $i = k$)

Round 2 (for P_i, $2 \leq i \leq k-1$):

1. Receive a set SS from P_{i-1}.
2. Encrypt SS with E_i.
3. Remove the first element from SS. (The first element $\{E^*_{i\sim 2,k\sim i+1}(S_i)\}$ will be fully encrypted.)
4. When $i < k-1$, send the first element to P_1 and the rest to P_{i+1}. When $i = k-1$, send the first element to P_1. (Note that, when $i = k-1$, the SS after taking out the first element will be empty, and thus P_k does not participate in Round 2.)

Table 2: The protocol for a passive party P_i.

However, though the initiator has access to the support vectors, they are fully encrypted. The initiator could simply inform the passive parties which of the data points it has are actually the support vectors and then get access to them. But this would violate our constraint that the initiator should learn nothing about the private data of any party. Thus, in order to perform testing/classification, we need to have an interactive protocol between all of the parties.

To perform testing securely, each passive party needs to evaluate the kernel functions on every pair of their testing data and the support vectors. A new instance can be treated the same as a test instance. To perform classification of a new instance, the kernel function must be evaluated for each pair of the new instance and support vector.

Such kernel functions can be represented as *another kernel matrix for testing* such that each element is the function output for a pair of testing data vector and support vector. Thus, we can use the secure protocol of computing the kernel matrix again: Suppose each party P_i holds a *testing set* S_i. The initiator P_1 starts the protocol by sending an *empty set* to P_2; In the same way as the protocol for training, P_1 will obtain the fully encrypted testing set from each party in two rounds. The descriptions for the two rounds are equivalent to those for training, i.e., Figures 3 and 4 and Table 1 and 2, except each set S_i here is a testing set. Then, P_1 can compute the testing kernel matrices for each party, as P_1 holds the fully encrypted support vectors. It can now go ahead and evaluate the kernel function.

Note that the initiator is prohibited from contributing to the testing set. This is to again prevent it from obtaining any linear equations. Thus, P_2 must reject P_1's start request if P_1 sends a non-empty set in the first round. No testing data nor linear equations will be disclosed to any party through this process.

4. EXPERIMENTAL RESULTS

In this section, we experimentally verify that our method is scalable to multiple parties in terms of the communication and computational time. The accuracy is equivalent to the case where the data is centralized. We used ten linux machines for our experiments, each with Pentium 4 2Ghz processors and 1 Gb memory.

Implementation

We use SVM-JAVA [15] - our own SVM implementation written in Java. SVM-JAVA is developed for research and educational purpose and implements the SMO (Sequential Minimal Optimization) algorithm for training SVM [23]. It currently supports linear and RBF kernels. Based on the SVM-JAVA, we implemented our privacy-preserving SVM classification solution using Java-RMI (Remote Method Invocation).

Datasets

We use the SPECT dataset [1] from the UCI machine learning repository. The SPECT dataset contains 80 data objects ($m = 80$) of 22 features ($n = 22$) and each object is represented as a binary feature vector; It has been shown that the preprocessed binary feature representation of the dataset performs better than the original continuous feature representation [1]. The SPECT dataset is not linearly separable and thus we used SVM RBF kernel (which is the most popularly used nonlinear kernel) and it generated high performance on cross validation. We also experimented on the tic-tac-toe dataset in the SMO package [11]. The tic-tac-toe dataset contains 958 data objects with 27 features.

Result

To check the scalability of the PP-SVM on a increasing number of participating parties, we vary the number of parties from three to ten. In each case, we randomly split the 80 data objects of the SPECT among the participating parties. For example, when six parties participated, four parties have 13 data objects and two parties have 14 data objects.

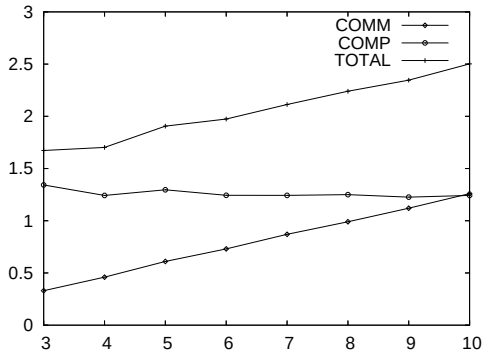

Figure 5: Training time on the SPECT dataset. X-axis: the number of participating parties; Y-axis: running time (sec.); COMM: communication time; TOTAL: total running time

Figure 5 shows the training time on the SPECT dataset for each number of participating parties. All the participating parties have the same result on the training set: The accuracy is 93.75%, and F1-score is 0.9367. The communication time and total training time increase linearly, as the number of participating parties increases. The computational time stays the same because the size of the kernel matrix at each party stays the same regardless of the number of parties. All the results are averaged over ten runs.

Figure 6 shows the training time on the tic-tac-toe dataset at each number of participating parties. All the parties also have the same result: The accuracy is 88.31% and F1 score is 0.9169. We see a

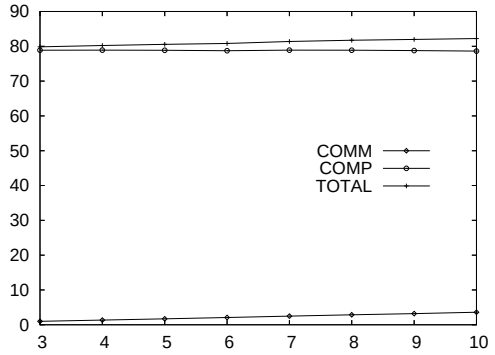

Figure 6: Training time on the tic-tac-toe dataset. X-axis: the number of participating parties; Y-axis: running time (sec.); COMM: communication time; TOTAL: total running time

similar result to that on the SPECT dataset: The communication time and total training time increase linearly while the computational time stays the same.

Note that the accuracy of our method is exactly the same as the accuracy of SVM on centralized data. Thus our method is suitable for any situation where using a SVM would be advisable.

5. RELATED WORK

Secure dot product computation

A secure dot product protocol could be used for the computation of the gram matrix. Several secure dot product algorithms have been recently developed [6, 27, 14]. However they are inefficient or not proven secure. For instance, the communication cost of the methods increases dramatically as the number of data items at each party increases; The communication cost becomes $O(m_1 * m_2)$ for two parties where m_1 and m_2 are the number of data for the two parties. Since they are not proven secure, parallelizing the dot product computations might cause a security breach.

The dot product computation method proposed in [24] is based on the stochastic estimates and thus produces inaccurate results especially when the data size is not large enough (e.g., < 1000). Our medical datasets in the experiment have only 80 data objects. A notable exception is the dot product protocol proposed by Goethals et al.[12] which is completely secure. This can actually be used. However, since this protocol requires homomorphic encryption (which is typically more inefficient), encryption of every element (0 or 1) in the vector, and decryption to get the final result, we believe our alternative is comparable, if not superior. However, where complete security is desired – one should use the dot product computation proposed by Goethals et al. as the basic sub-protocol for the PP-SVM computation. Our key goal here is simply to show that privacy-preserving SVM is feasible and practical.

Privacy-preserving data mining and SVM

Recently, there has been significant interest in the area of Privacy-Preserving Data Mining. We briefly cover some of the relevant work. Several solution approaches have been suggested. One approach is to perturb the local data (by adding "noise") before the data mining process, and mitigate the impact of the noise from the data mining results by using reconstruction techniques [3, 2, 9, 25]. However, there is some debate about the security properties of such algorithms [17].

The alternative approach of using cryptographic techniques to protect privacy was first addressed for the construction of decision trees [20, 21]. Our work follows the same approach. Prior work has also addressed association rule mining [31, 16], clustering [19, 28], and classification [7, 29, 33].

Yu and Vaidya [36] develop a privacy preserving linear SVM classification algorithm. Since their method is based on the optimization formulation of the proximal SVM [10], it is limited to linear classification. Privacy preserving data mining on *vertically partitioned* data has also been looked at[26]. Yu and Vaidya [37] propose a privacy preserving SVM on *vertically partitioned data*. This is complementary to our work. This paper proposes a method for SVM *nonlinear kernels* and is applicable to only *horizontally partitioned data*.

6. CONCLUSION

This paper proposes a privacy-preserving solution for SVM *nonlinear* classification on *horizontally partitioned data*. The global SVM classification model is securely constructed from the data distributed at multiple parties. Our method runs with the data represented by binary feature vectors and uses the secure set intersection cardinality to securely compute the gram matrix. The independent intermediator computes the kernel matrix and then the global SVM model from the gram matrix. No data is disclosed to it. Our protocol is secure and efficient: the communication cost is linear to the number of participating parties. Our key result has been to show that privacy-preserving support vector machine classification is feasible and efficient. However, avenues for future work remain. Though appropriate for many real life situations, the requirement of an untrusted intermediator seems overly restrictive. A key challenge for the future is to remove the need of the intermediator, while still efficiently constructing the SVM.

7. REFERENCES

[1] SPECT dataset. ftp://ftp/ics.uci.edu/pub/machine-learning-databases/spect/.

[2] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 247–255, 2001.

[3] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD Conference on Management of Data*, pages 439–450, 2000.

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

[5] N. Christianini and J. Shawe-Taylor. *An Introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[6] W. Du and M. J. Atallah. Privacy-preserving statistical analysis. In *Proceeding of the 17th Annual Computer Security Applications Conference*, 2001.

[7] W. Du and Z. Zhan. Building decision tree classifier on private data. In *IEEE International Conference on Data Mining Workshop on Privacy, Security, and Data Mining*, pages 1–8, 2002.

[8] Directive 95/46/EC of the european parliament and of the council of 24 october 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. *Official Journal of the European Communities*, No I.(281):31–50, Oct. 24 1995.

[9] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–228, 2002.

[10] G. Fung and O. L. Mangasarian. Proximal support vector machine classifiers. In *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD'01)*, pages 77–86, 2001.

[11] X. Ge. C++ code: SMO training of SVM. http://www.datalab.uci.edu/people/xge/svm/, 2001.

[12] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikäinen. On Secure Scalar Product Computation for Privacy-Preserving Data Mining. In C. Park and S. Chee, editors, *The 7th Annual International Conference in Information Security and Cryptology (ICISC 2004)*, volume 3506, pages 104–120, December 2–3, 2004.

[13] Standard for privacy of individually identifiable health information. *Federal Register*, 66(40), Feb. 28 2001.

[14] I. Ioannidis, A. Grama, and M. Atallah. A secure protocol for computing dot-products in clustered and distributed environments. In *The 2002 International Conference on Parallel Processing*, 2002.

[15] X. Jiang and H. Yu. SVM-JAVA: A Java implementation of the SMO (sequential minimal optimization) for training SVM. Computer Science Department, University of Iowa, http://hwanjoyu.org/svm-java, 2005.

[16] M. Kantarcıoğlu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1026–1037, Sept. 2004.

[17] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM'03)*, 2003.

[18] L. A. Kurgan, K. J. Cios, R. Tadeusiewicz, M. Ogiela, and L. S. Goodenday. Knowledge discovery approach to automated cardiac spect diagnosis. *Artificial Intelligence in Medicine*, 23:2:149–169, 2001.

[19] X. Lin, C. Clifton, and M. Zhu. Privacy preserving clustering with distributed EM mixture modeling. *Knowledge and Information Systems*, to appear 2004.

[20] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Advances in Cryptology – CRYPTO 2000*, pages 36–54. Springer-Verlag, Aug. 20-24 2000.

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

[22] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Oct. 1996.

[23] J. Platt. Fast training of support vector machines using sequential minimal optimization. In A. S. B. Scholkopf, C. Burges, editor, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1998.

[24] P. Ravikumar, W. W. Cohen, and S. E. Fienberg. A secure protocol for computing string distance metrics. In *Proc. the Workshop on Privacy and Security Aspects of Data Mining at the Int. Conf. on Data Mining*, 2004.

[25] S. J. Rizvi and J. R. Haritsa. Maintaining data privacy in association rule mining. In *Proceedings of 28th International Conference on Very Large Data Bases*, pages 682–693, Hong Kong, Aug. 20-23 2002. VLDB.

[26] J. Vaidya. *Privacy Preserving Data Mining over Vertically Partitioned Data*. PhD thesis, Purdue University, West Lafayette, Indiana, 2004.

[27] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 639–644, 2002.

[28] J. Vaidya and C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 206–215, 2003.

[29] J. Vaidya and C. Clifton. Privacy preserving naïve bayes classifier for vertically partitioned data. In *2004 SIAM International Conference on Data Mining*, pages 522–526, 2004.

[30] J. Vaidya and C. Clifton. Secure set intersection cardinality with application to association rule mining. *Journal of Computer Security*, 2005.

[31] J. Vaidya and C. Clifton. Secure set intersection cardinality with application to association rule mining. *Journal of Computer Security*, 13(4), Nov. 2005.

[32] V. N. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, 1998.

[33] R. Wright and Z. Yang. Privacy-preserving bayesian network structure computation on distributed heterogeneous data. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Seattle, WA, Aug.22-25 2004.

[34] X. Yan and J. Han. Closegraph: Mining closed frequent graph patterns. In *Proc. Int. Conf. Knowledge Discovery and Data Mining (KDD'03)*, 2003.

[35] H. Yu, K. C. Chang, and J. Han. Heterogeneous learner for Web page classification. In *Int. Conf. Data Mining (ICDM'02)*, 2002.

[36] H. Yu and J. Vaidya. Privacy-preserving linear SVM classification. *Submitted for publication*, 2005.

[37] H. Yu, J. Vaidya, and X. Jiang. Privacy preserving svm classification on vertically partitioned data. *Submitted for publication*, 2005.