Knowl Inf Syst (2008) 14: 161–178
DOI 10.1007/s10115-007-0073-7

Knowledge and Information Systems

REGULAR PAPER

Jaideep Vaidya · Hwanjo Yu · Xiaoqian Jiang

Privacy-preserving SVM classification

Received: 12 April 2006 / Revised: 14 November 2006 / Accepted: 26 January 2007
Published online: 24 March 2007

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 is required is distributed knowledge discovery that is sensitive to this problem. The key is to obtain valid results, while providing guarantees on the nondisclosure 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 data distributed at multiple parties, without disclosing the data of each party to others. Solutions are sketched out for data that is vertically, horizontally, or even arbitrarily partitioned. We quantify the security and efficiency of the proposed method, and highlight future challenges.

Keywords Support vector machine · Classification · Privacy · Security

1 Introduction

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

J. Vaidya (✉)
Management Science and Information Systems Department, Rutgers University, Newark, NJ 07102, USA
E-mail: jsvaidya@rbs.rutgers.edu

H. Yu · X. Jiang
Department of Computer Science, University of Iowa, Iowa City, IA, USA

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 example, HIPAA laws [32] 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 has proven to be effective in many real-world applications [6, 7, 41]. 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 may be partitioned horizontally, vertically, or in an arbitrary manner between the 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 many different contexts. For example, consider a clearing house for a consortium of banks. The 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. Thus, the data is horizontally distributed. 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.

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.

Our method is based on gram matrix computation. A variety of different kernels can be computed from the gram matrix, both linear and nonlinear. 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. 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 noncollusion is quite strict, it enables an efficient algorithm. We later discuss this issue in more detail.

The rest of the paper is organized as follows. In Sect. 2, we overview Support Vector Machines. In Sect. 3, we develop our PP-SVM technique for vertically partitioned data. In Sect. 4, we briefly sketch the solution for horizontally partitioned data. Finally, in Sect. 5, we develop the solution for arbitrarily partitioned data. Related work is discussed in Sect. 6 and Sect. 7 concludes the paper.

2 SVM overview

In this section, we first describe the notation and give an overview of support vector machines.

Notation

All vectors are column vectors unless transposed to a row vector by a prime superscript $'$. The scalar (inner) product of two vectors x and y in the n-dimensional real space R^n is denoted by $x'y$ 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 ith 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 $\mathcal{I}$.

2.1 SVM overview

First, consider a linear binary classification task, as depicted in Fig. 1. For this problem, SVM finds the separating hyperplane ($w \cdot x = \gamma$) that maximizes the *margin*, denoting the distance between the hyperplane and closest data points (i.e., support vectors). In practice, we use the "soft" margin to deal with noise, in which the distance from the boundary to each support vector could be different. The "hard" margin is formulated as $\frac{1}{||w||}$, as illustrated in Fig. 1. To maximize the margin while minimizing the error, the standard SVM solution is formulated into the following primal program [12, 41]:

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

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

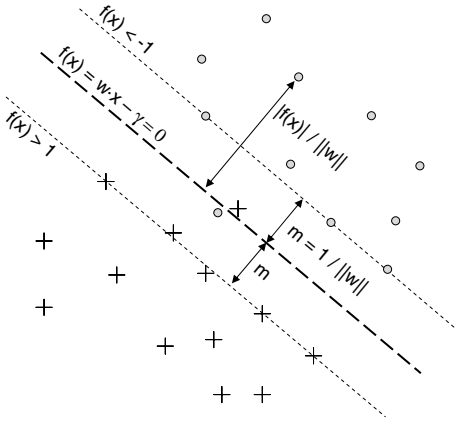

Fig. 1 The separating hyperplane that maximizes the margin. ("+" is a positive data point, i.e., $f(\text{"+"}) > 0$, and "∘" is a negative data point, i.e., $f(\text{"}\circ\text{"}) < 0$)

which minimizes the reciprocal of the margin (i.e., $w'w$) and the error (i.e., $e'y$). By having the slack variable y in the constraint (2), SVM allows error or the soft margin. The slack or error is minimized in the objective function (1) and it will be larger than zero when the point is on the wrong side or within the margin area. The soft margin parameter ν (a user parameter) 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. Once w and γ are computed, we can determine the class of a new data object x by $f(x) = w'x - \gamma$, where the class is *positive* if $f(x) > 0$, or else *negative*.

In order to reduce the number of variables in the objective function and also be able to apply the kernel trick, we transform the primal problem to the following dual problem by applying the Largrange multipliers:

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

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

where $d_i (i.e., \mathcal{D}_{ii})$ and α_i are the class label and the coefficient, respectively, for a data vector x_i. The coefficients α are to be computed from this dual problem. An $m \times m$ matrix $\mathcal{Q}$ is computed by the scalar product of every data pair, i.e., $\mathcal{Q}_{ij} = K(x_i, x_j) d_i d_j$ where $K(x_i, x_j) = x_i \cdot x_j$ for linear SVM. The support vectors are the data vectors $\{x_i\}$ such that the corresponding coefficients $\alpha_i > 0$. The weight vector $w = \sum \alpha_i d_i x_i$ and thus the classification function $f(x) = \sum \alpha_i d_i x_i \cdot x - \gamma$ 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, $K(x_i, x) = (x_i \cdot x + 1)^p$ for polynomial kernel,). Reference [41] provides further details on SVM.

3 Privacy-preserving SVM over vertically partitioned data

When data is vertically partitioned, parties collect different information about the same set of entities. For instance, a bank, health insurance company and auto insurance company collect different information about the same people. A bank has customer information like average monthly deposit, account balance, etc. The health insurance company has access to medical information and other policy information. The car insurance company has access to information such as car type, accident claims, etc. Together, they might evaluate if the person is a credit risk for life insurance.

As we see from the last paragraph of Sect. 2, an SVM model is represented by the bias γ, and a list of support vectors, their labels, and coefficients $\{(x_i, d_i, \alpha_i)\}$ such that $\alpha_i > 0$. That is, the global model $\mathcal{G}$ is composed of γ and $\{(x_i, d_i, \alpha_i)\}$ which are computed from the dual problem in Sect. 2.

Given vertically partitioned data over multiple parties, we cannot use a local SVM model (i.e., computed only over local data), because the global SVM model $\mathcal{G}$ cannot be built only from local SVM models; The globally optimal coefficients (computed by the dual problem) will be different from the locally optimal coefficients computed on local data. Since each party has the data of an attribute subset, the dual problem on the attribute subset will not generate the globally optimal coefficients.

To solve the dual problem globally, we need the $m \times m$ matrix $\mathcal{Q} = K(x_i, x_j)d_i d_j$ in Eq. (3) which is computed over the data of all the attributes. The diagonal matrix $\mathcal{D}$ for d_i is given as class labels, thus we only need to compute the global kernel matrix $\mathcal{K} = K(x_i, x_j)$. For linear kernel where $K(x_i, x_j) = x_i \cdot x_j$, the global matrix $\mathcal{K}$ can be directly computed from local matrices because $\mathcal{K}$ is a gram matrix and a gram matrix can be merged from gram matrices of vertically partitioned data, as Lemma 1 proves.

Lemma 1 *Suppose the $m \times n$ data matrix $\mathcal{A}$ is vertically partitioned into $\mathcal{A}^1$ and $\mathcal{A}^2$ as Fig. 2 illustrates. Let $\mathcal{K}^1$ and $\mathcal{K}^2$ be the $m \times m$ gram matrices of matrices $\mathcal{A}^1$ and $\mathcal{A}^2$, respectively. That is, $\mathcal{K}^1 = \mathcal{A}^1\mathcal{A}^{1'}$ and $\mathcal{K}^2 = \mathcal{A}^2\mathcal{A}^{2'}$. Then, $\mathcal{K}$, the gram matrix of $\mathcal{A}$, can be computed as follows:*

$$\mathcal{K} = \mathcal{K}^1 + \mathcal{K}^2 = \mathcal{A}^1\mathcal{A}^{1'} + \mathcal{A}^2\mathcal{A}^{2'} \tag{5}$$

Proof An (i, j)th element of $\mathcal{K}$ is $x_i \cdot x_j$, where x_i and x_j are ith and jth data vectors in $\mathcal{A}$. Let x_i^1 and x_i^2 be vertically partitioned vectors of x_i, which are the

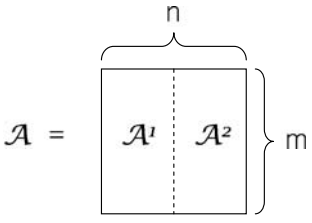

Fig. 2 Vertically partitioned matrix $\mathcal{A}$

parts from $\mathcal{A}^1$ and $\mathcal{A}^2$, respectively. Then,

$$x_i \cdot x_j = x_i^1 \cdot x_j^1 + x_i^2 \cdot x_j^2. \tag{6}$$

From Eq. (6), each element in $\mathcal{K}$ is equal to the sum of the elements in $\mathcal{K}^1$ and $\mathcal{K}^2$. Thus $\mathcal{K} = \mathcal{K}^1 + \mathcal{K}^2$. □

Lemma 1 proves that local gram matrices are sufficient to build the global gram matrix which is the kernel matrix $\mathcal{K}$ for linear kernel. Some popular nonlinear kernel matrices can also be computed from the gram matrix: The polynomial kernel is represented by a dot product of data vectors (i.e., $K(x_i, x_j) = (x_i \cdot x_j + 1)^p$). The RBF kernel can also be represented by dot products (i.e., $K(x_i, x_j) = \exp(-\frac{||x_i - x_j||^2}{g}) = \exp(-\frac{|x_i \cdot x_i - 2x_i \cdot x_j + x_j \cdot x_j|}{g})$). Thus, the local gram matrix from each party is sufficient to construct the global kernel matrix $\mathcal{K}$ for nonlinear kernels such as polynomial and RBF that can be represented by dot products.

Once the global gram matrix is built from the local gram matrices, each party can run a quadratic programming solver to compute the global SVM model $\mathcal{G}$, which will be the same for every party. If the global gram matrix is computed at an independent site, that site then computes the global model.

3.1 Secure merge of local models

In order to securely merge the local $m \times m$ gram matrices, a *secure addition* mechanism for $m \times m$ matrices is required. For $k \geq 3$ parties, we present such a method based on simple secure addition of scalars.

We first describe a simple method to securely calculate the sum of integers from individual sites under the assumption that there are at least three parties and the parties do not collude. We then extend the method so as to seamlessly merge the local models with high efficiency and privacy.

3.1.1 Secure sum of integers

Formally, we assume $k \geq 3$ parties, $P_0, \ldots, P_{k-1}$, with party P_i holding value v_i. Together, they want to compute the sum $v = \sum_{i=0}^{k-1} v_i$. Assume that the sum v is known to lie in a field $\mathcal{F}$.

The parties also randomly order themselves into a ring. The ordering can be selected by one of the parties, or by a third party. If the parties cannot decide on a suitable order and no third party can be found, then a protocol developed by Sweeney and Shamos can be used [33] to fix upon a random ordering. The protocol developed by Sweeney and Shamos is quite efficient and requires only $O(k)$ communication. For this paper, to simplify the presentation, without loss of generality, we assume that this order is the canonical order $P_0, \ldots, P_{k-1}$. In general, any order can be decided on. The protocol proceeds as follows:

P_0 randomly chooses a number R, from a uniform distribution over $\mathcal{F}$. P_0 adds this to its local value v_0, and sends the sum $R + v_0 \bmod |\mathcal{F}|$ to site P_1. For the remaining sites $P_i, i = 1, \ldots, k-1$, the algorithm proceeds as follows:

P_i receives

$$V = R + \sum_{j=0}^{i-1} v_j \bmod |\mathcal{F}|.$$

P_i then computes

$$R + \sum_{j=1}^{i} v_i \bmod |\mathcal{F}| = (v_i + V) \bmod |\mathcal{F}|$$

and passes it to site $P_{i+1 \pmod k}$. Finally, P_0, subtracts R from the final message it gets (i.e., adds $-R \pmod{|\mathcal{F}|}$) to compute the actual result.

Clearly, the above protocol correctly calculates the required sum. In order to evaluate the security of the protocol, it is necessary to have a definition of *what* is meant by security. The area of Secure Multiparty Computation (SMC) provides a theoretical framework for defining and evaluating secure computation. This protocol can be proven to be completely secure under our assumptions in the SMC framework. A complete proof of security is presented in our technical report [47].

3.1.2 Secure sum of matrices

We can extend the secure addition of scalars to securely adding matrices. The key idea is as follows. Suppose a master party wants to merge (i.e., add) its local matrix with those in other slave parties. We assume that the parties have arranged themselves in some sequence and the master initiates the protocol.

1. The master party creates a random matrix of the same size as its local matrix. (The random matrix is hidden from the other parties.)
2. The master party merges (adds) the random matrix with its local matrix, sends the merged matrix to the following slave party.
3. Each slave party, receives the perturbed matrix, merges it with its local matrix and passes it to the following party (the last slave party sends the matrix back to the master).
4. The master subtracts the random matrix from the received matrix, which results in the matrix that adds the matrices of all the parties, without disclosing their local matrices to each other.

All addition is done in a closed field, and subtraction refers to addition of the complement. This secure addition mechanism is proven to be secure and efficient [47, 49]. The extra computation required by the first party is the generation of the random matrix and the final subtraction. In terms of communication overhead, k rounds are required for every party to acquire the summed matrix, where k is the number of participating parties. One problem with the matrix summation method is that it is vulnerable to collusion. The parties preceding and following a party, can collude to recover its local matrix. However, the technique can easily be made collusion resistant to q parties by splitting up the local matrices into q random parts and carrying out the addition protocol q times. The sum of the final matrices from all q rounds gives the real global matrix. As long as the parties are ordered differently in each run, recovery of a local matrix is only possible if collusion occurs in all q rounds. Further details can be found in [47].

3.2 Security

Our method preserves "data privacy", since only the original party gets to exactly see the data. The local model is directly computed from the local data. However, to ensure "model privacy," we need at least three participating parties. Each party gets the final global model, which is simply the sum of local models. Thus, with only two parties participating, the other party's local matrix could be found simply by subtracting the local model from the global model. What is revealed, is the sum of the local models of the other parties. Since the SVM requires knowing the global matrix, this is always possible from the final result as well, and so is unavoidable.

We still need to analyze the effects of knowing the sum of gram matrices computed over the attributes of other parties. In general, the number and type of attributes of the other parties are still assumed to be unknown. As such, the summed matrix does not disclose any attribute values. If the exact number and types of attributes of the other parties are known, a number of quadratic equations will be revealed in the attribute values; Every cell of the gram matrix corresponds to a dot product—thus the quadratic equation. Since the matrix is symmetric, there are a total of $m(m+1)/2$ distinct equations (where m is the number of data objects). If the number of total variables (i.e., the sum of all the attributes of other parties) is larger than $m(m+1)/2$, it is impossible to recover the exact attribute values. Knowing that the matrix is symmetric and positive semidefinite does not disclose further information. While this does reveal more information than strictly necessary, this is a tradeoff in the favor of efficiency. If complete security is required, the summed matrix could be kept randomly split between two of the parties, and an oblivious protocol run to compute the global model using the generic circuit evaluation technique developed for secure multiparty computation [14, 45]. Otherwise, if a third party holds the final model and is asked to do new classfications, all leakage is avoided (since the third party has no extra information).

3.3 Testing

Once the global SVM model is constructed at each party, to test a new data object using the model, we need to compute dot products between the support vectors and the new data object. The testing function is formulated as $f(x) = \sum \alpha_i d_i K(x_i, x) - \gamma$ where x_i is a support vector and x is a testing data object. After training, each party holds coefficients α, label d, bias γ, and partial attributes of the support vectors and the testing data object. Thus, from Eq. (6), to securely compute the dot products and thus the kernel functions between the support vectors and the testing data object, we need to run the secure addition of vectors, such that each element of the vector is a dot product between a partial support vector and the testing data object. For instance, when each party holds m' partial support vectors, the vector needs to be securely merged (or added) will have m' elements. Its security proof will be the same as that for the secure addition of matrices, as a column vector is a matrix of one column. Accordingly, to test a new data object, one round of communication is needed.

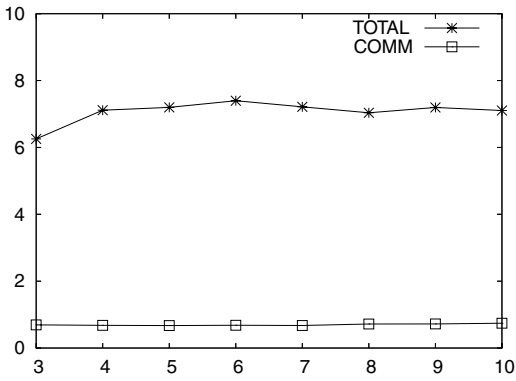

Fig. 3 *X*-axis: # parties; *Y*-axis: time (s.); COMM: communication time; TOTAL: total training time

3.4 Experimental evaluation

The goal of the experiments is simply to demonstrate the scalability of our PP-SVMV. The accuracy will be exactly the same as that of SVM when the data is centralized. We revised the sequential minimal optimization (SMO) source[1] to implement the PP-SVMV. We used the Tic-Tac-Toe data set included in the SMO package for our experiment. We sampled around 958 data objects (m) and extracted around 27 features (n). PP-SVMV generates above 99% with an RBF kernel that is the same as that of the original SVM when the data is centralized.

To check the scalability of the PP-SVMV on an increasing number of participating parties, we vary the number of parties from three to 10. We divide the 27 features about equally between the participating parties. For instance, when 10 parties participate, three parties have two features, and the other seven parties have three features. Figure 3 shows results of our experiments: The total training time (including the parallel local computations) hardly changes; SVM is sensitive to the number of data objects more than the features, and the change in the number of features is not visibly influential to the total training time. The difference of the communication time is also not visible due to the dominant computation time. The results are averaged over 10 runs.

4 Privacy-preserving SVM over horizontally partitioned data

With horizontal partitioning of data, each party collects the same features of information for different data objects. 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.

Thus, in this 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.

[1] http://www.datalab.uci.edu/people/xge/svm

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 Eqs. (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* $G_{ij} = x_i \cdot x_j$. As stated before, the nonlinear kernel matrices like polynomial or RBF can be computed from the gram matrix $\mathcal{G}$.

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 [9, 16, 29, 34] are inefficient or insecure to be applied for computing the gram matrix. The scalar product protocol of Goethals et al. [13] is completely secure and can indeed be used. One simply needs to run the protocol for each data pair to compute each scalar product. Indeed, a modified version of this is used to compute the gram matrix from arbitrarily partitioned data, as described in Sect. 5 later.

However, as proposed in Yu et al. [46], with binary feature vectors a more communication-efficient alternative is possible. This is based on using a secure protocol [39] for computing the size of the intersection of local sets.

To use the secure set intersection cardinality [39], 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 a commutative one-way hash function. The key idea here is that the commutative property guarantees that the order of hashing does not matter. Thus, each party hashes its set of values and passes them on to the next party to hash and send forward. Now, the same original entries will overall hash to the same values, while the one-way property guarantees that a hashed value cannot be used to infer the original. More details on this can be found in [39]. However, 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 [39], 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. Yu et al. [46] presents the complete details for the scalable and secure protocol that computes the kernel matrix in linear time, without disclosing any data.

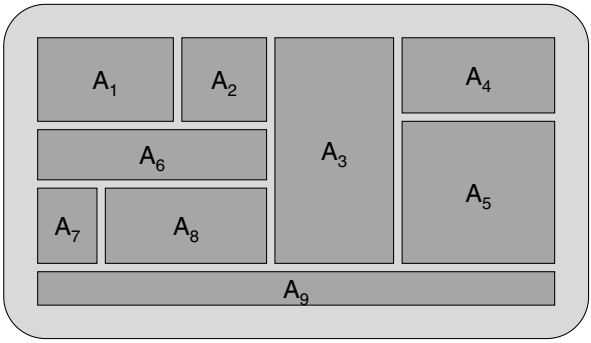

Fig. 4 A data set split between nine parties

5 Privacy-preserving SVM over arbitrarily partitioned data

Both vertical partitioning and horizontal partitioning are special cases of arbitrary data partitioning. In arbitrary data partitioning, different parties may own arbitrary subsets of the data. For example, Fig. 4 depicts an example where nine parties jointly hold the complete data set (data set A_i is held by party P_i).

Though, completely arbitrary partitioning of data is unlikely in practice, a general solution for this case would work for *any* data partitioning. Thus, such a general solution will be of great use. We now show how to generate the gram matrix when the data is arbitrarily partitioned. The key idea is to generate the gram matrix, one element at a time, using a modified version of the scalar product protocol of Goethals et al. [13], based on homomorphic encryption. Homomorphic Encryption is a semantically-secure public-key encryption which, in addition to standard guarantees has the additional property that given any two encryptions $E(A)$ and $E(B)$, there exists an encryption $E(A * B)$ such that $E(A) * E(B) = E(A * B)$, where $*$ is either addition or multiplication (in some abelian group). The cryptosystems mentioned earlier are additively homomorphic (thus the operation $*$ denotes addition). Using such a system, it is quite simple to create a scalar product protocol. The key is to note that $\sum_{i=1}^{n} x_i \cdot y_i = \sum_{i=1}^{n} (x_i + x_i + \cdots + x_i)$ (y_i times). Since each vector is arbitrarily partitioned, the party owning each x_i encrypts and sends it to the party owning the corresponding y_i. This party can now use the additive homomorphic property to compute the product in encrypted form. Again, using the additive property the sum of all the products can be computed to compute the dot product. The specific details are given in Algorithm 1. There are several homorphic encryption systems such as the Goldwasser–Micali cryptosystem [5], the Benaloh cryptosystem [4], the Naccache–Stern cryptosystem[25], the Paillier cryptosystem [28], and the Okamoto–Uchiyama cryptosystem [26], any of which can be used in practice.

5.1 Security

Theorem 1 *In the absence of collusion, Algorithm 1 is secure in the semihonest model—only Q learns the gram matrix while the other parties learn nothing.*

Algorithm 1 Computing the gram matrix from arbitrarily partitioned data

Require: Total m data points and n features split in some arbitrary fashion between k parties
Require: Data represented by matrix A; A_{bc} represents the value of the c^{th} feature of the b^{th} point
Require: A third party Q, who receives the gram matrix and creates the classifier
1: Q creates a new semantically secure homomorphic encryption system keypair $\{pk, sk\}$
2: Q sends the public key pk to all of the parties
3: **for** $i = 1 \ldots m$ **do**
4: **for** $j = 1 \ldots m$ **do**
5: {Compute the dot product of data point i with data point j}
6: **for** $k = 1 \ldots n$ **do**
7: Let P_a hold A_{ik} and P_b hold A_{jk}
8: P_a computes $m_k = E_{pk}(A_{ik}, r)$, where r is a random nonce and sends it to P_b
9: P_b computes $m'_k = m^{A_{jk}} = E_{pk}(A_{ik}, r)^{A_{jk}} = E_{pk}(A_{ik} * A_{jk}, r')$ where r' is some number from the domain of r
10: **end for**
11: {The parties together compute $\prod_{k=1}^{n} m'_k$}
12: res $= 1$
13: **for** $k = 1 \ldots n - 1$ **do**
14: The party that owns m'_k computes $res = res * m'_k$ and sends it to the party owning m'_{k+1}
15: **end for**
16: The party owning m'_n computes $res = res * m'_n$ and sends it to Q
17: Q receives $res = \prod_{k=1}^{n} m'_k = E_{pk}(\sum_{k=1}^{n} A_{ik} * A_{jk}, r'')$
18: Q decrypts this using sk to get the desired dot product
19: **end for**
20: **end for**

Proof The first message sent by Q is simply the encryption key chosen—this can be simulated by each party simply by choosing a random key. After this, every message received by any party is encrypted by ek. Since the cryptosystem used is semantically secure, and the decryption key is known only to Q, each message can be simulated using a random number. Finally, Q gets to know the gram matrix as the final result, so it can easily simulate the messages sent to it. □

Collusion with Q causes problems—a party could collude with Q to reveal the other party's input. Though the assumption of noncollusion is often implicitly made in the real world, this is a serious problem. Even if it were possible to find trustworthy real-world relationships, technical solutions would be more satisfying. We now present a modification to the algorithm to make it collusion resistant. In lines 8–10, instead of using the public key sent by Q, P_a creates its own keypair and uses it to encrypt. Now it is no longer possible for P_b to collude with Q to reveal P_a's input. To protect P_b, instead of sending back the encryption of $(A_{ik} * A_{jk})$, a share of it is sent back. This at the end of line 10, after decryption, both P_a and P_b have shares of $(A_{ik} * A_{jk})$. Now, all the parties can together run the collusion-resistant secure sum protocol to finally compute the dot product. In this way, each party is protected against collusion to the extent of the collusion resistance of the secure sum protocol.

Table 1 Estimated time for computing the gram matrix

t	m	n	Estimated time
512	100	10	2.5 min
512	200	10	10 min
512	500	20	2.08 h
512	1000	30	12.5 h
1024	100	10	17.33 min
1024	200	10	1.15 h
1024	500	20	14.44 h
1024	1000	30	86.66 h

5.2 Computation and communication cost

The computation and communication cost of this algorithm is quite reasonable. We next sketch out the worst case estimate of time required, when the data is completely fragmented between mn parties, each owning exactly one element of one point. Here, each scalar product requires n encryptions, n exponentiations, n multiplications, and 1 decryption. There are a total of m^2 scalar products. Thus, a total of m^2n encryptions, m^2n exponentiations, m^2n multiplications, and m^2 decryptions are required to compute the gram matrix. The cost of the exponentiations dominates all of the other costs. One exponentiation requires 0.0015 s for 512 bit encryption and $0.0104s$ for 1024 bit encryption on an Intel Pentium Dual 830, 3.00 GHZ with 2 GB RAM. Table 1 gives the estimated approximate cost for several different values of m and n. Again, note that these are the worst case times, when the entire data matrix is split between different parties. In reality, the time taken will be orders of magnitude lower depending on how many scalar products or elements of it can be done locally. In the case of pure vertical partitioning, we have already seen that a dataset of about 1000 points with about 30 different features, requires about 10 s. Similarly, with pure horizontal partitioning, the efficient protocol in Yu et al. [46] only requires about 80 s. While the arbitrary partitioning protocol will correspondingly take more time, it is still significantly less than the maximum times given later (for completely partitioned data).

6 Related work

Recently, there has been significant interest in the area of Privacy-Preserving Data Mining. We briefly cover some of the relevant work. Work in PPDM has followed two major directions—the randomization/perturbation approach and the cryptographic approach.

In the perturbation approach, data is locally perturbed by adding "noise" before mining is done. For example, if we add a random number chosen from a Gaussian distribution to the real data value, the data miner no longer knows the exact value. However, important statistics on the collection (e.g., average) will be preserved. Special techniques are used to reconstruct the original distribution (not the actual data values). The mining algorithm is modified to work while taking this into consideration. The seminal paper by Agrawal and Srikant

[3] introduced this notion to the data mining community. Several different algorithms are proposed to reconstruct distributions and learn a decision tree classifier from the perturbed data. A convergence result was proved in [1] for a refinement of this algorithm. There has been work using the same approach for association rule mining [11, 30, 50]. Zhu and Lei [51] study the problem of optimal randomization for privacy-preserving data mining and demonstrate the construction of optimal randomization schemes for density estimation. This model works in the "data warehouse" model of data mining, but trades off privacy for accuracy of results. Xu et al. [44] have developed a sparsified singular value decomposition method for data distortion to preserve privacy. However, several problems have been pointed out with the privacy inherent in such an approach [15, 19, 20, 24].

The alternative approach of using cryptographic techniques to protect privacy was first utilized for the construction of decision trees [23]. There has since been significant work on many techniques in data mining. Specifically, secure methods have been proposed for association rule mining [18, 39], clustering [17, 22, 35], classification [10, 36, 38, 43], outlier detection [37], and regression [21, 31]. Our work follows the same approach. A good overview of prior work in this area can be found in [40, 42]. Recently, some alternative techniques such as condensation [2] and transformation [27] have also been proposed, though the security properties of these need to be carefully investigated.

All of the cryptographic work falls under the theoretical framework of Secure Multiparty Computation. Yao first postulated the two-party comparison problem (Yao's Millionaire Protocol) and developed a provably secure solution [45]. This was extended to multiparty computations by Goldreich et al. [14]. The key result in this field is that *any* function can be computed securely. Thus, the generic circuit evaluation technique can be used to solve our current problem. However, the key issue in privacy-preserving data mining is one of efficiency. The generic technique is simply not efficient enough for large quantities of data. This paper proposes an efficient technique to solve the problem.

Yu and Vaidya [49] developed a privacy-preserving SVM classification on *horizontally partitioned* data. Since their method is based on the trick of the proximal SVM [12], it is limited to linear classification. Yu et al. [46] proposes a technique for securely computing nonlinear kernels as well. Reference [48] tackles the problem when the data is vertically partitioned. In this paper, we integrate the known approaches and propose secure techniques that work when data is arbitrarily partitioned as well.

7 Conclusion

We propose a scalable solution for privacy-preserving SVM classification based on gram matrix computation. Assuming an independent untrusted third party, we show how to securely compute the global SVM model, without disclosing any extra information about the data of each party to others. Future work may address the idea of efficiently achieving complete security by keeping the global model split between parties as well.

References

1. Agrawal D, Aggarwal CC (2001) 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. ACM, Santa Barbara, CA, pp 247–255. [Online]. Available: http://doi.acm.org/10.1145/375551.375602
2. Aggarwal CC, Yu PS (2004) A condensation approach to privacy preserving data mining. In: Lecture notes in computer science, vol 2992, pp 183–199
3. Agrawal R, Srikant R (2000) Privacy-preserving data mining. In: Proceedings of the 2000 ACM SIGMOD conference on management of data, ACM, Dallas, TX, pp 439–450. [Online]. Available: http://doi.acm.org/10.1145/342009.335438
4. Benaloh JC (1986) Secret sharing homomorphisms: Keeping shares of a secret secret. In: Odlyzko A (ed) Advances in cryptography—CRYPTO86: proceedings, vol 263, Lecture notes in computer science, Springer-Verlag, Berlin, pp 251–260. [Online]. Available: http://springerlink.metapress.com/openurl.asp?genre=article&issn=0302-9743&volume=263&spage=251
5. Blum M, Goldwasser S (1984) An efficient probabilistic public-key encryption that hides all partial information. In: Blakely R (ed) Advances in cryptology—Crypto 84 proceedings. Springer-Verlag, Berlin
6. Burges CJC (1998) A tutorial on support vector machines for pattern recognition. Data Min Knowl Discov 2:121–167
7. Christianini N, Shawe-Taylor J (2000) An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press, New York
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. Off J Eur Communities I(281):31–50
9. Du W, Atallah MJ (2001) Privacy-preserving statistical analysis. In: Proceedings of the 17th annual computer security applications conference, New Orleans, LA [Online]. Available: http://www.cerias.purdue.edu/homes/duw/research/paper/acsac2001.ps
10. Du W, Zhan Z (2002) Building decision tree classifier on private data. In: Clifton C, Estivill-Castro V (eds) IEEE international conference on data mining workshop on privacy, security, and data mining, vol 14. Australian Computer Society, Maebashi City, Japan, pp 1–8. [Online]. Available: http://crpit.com/Vol14.html
11. Evfimievski A, Srikant R, Agrawal R, Gehrke J (2002) Privacy preserving mining of association rules. In: The eighth ACM SIGKDD international conference on knowledge discovery and data mining, Edmonton, Alberta, Canada, pp 217–228. [Online]. Available: http://doi.acm.org/10.1145/775047.775080
12. Fung G, Mangasarian OL (2001) Proximal support vector machine classifiers. In: Proceedings of the ACM SIGKDD international conference knowledge discovery and data mining (KDD'01), pp 77–86
13. Goethals B, Laur S, Lipmaa H, Mielikäinen T (2004) On secure scalar product computation for privacy-preserving data mining. In: Park C, Chee S (eds) The 7th annual international conference in information security and cryptology (ICISC 2004), vol 3506, pp 104–120
14. Goldreich O, Micali S, Wigderson A (1987) How to play any mental game—a completeness theorem for protocols with honest majority. In: 19th ACM symposium on the theory of Computing, pp 218–229. [Online]. Available: http://doi.acm.org/10.1145/28395.28420
15. Huang Z, Du W, Chen B (2005) Deriving private information from randomized data. In: Proceedings of the 2005 ACM SIGMOD international conference on management of data, Baltimore, MD
16. Ioannidis I, Grama A, Atallah M (2002) A secure protocol for computing dot-products in clustered and distributed environments. In: The 2002 international conference on parallel processing, Vancouver, British Columbia, Canada
17. Jagannathan G, Wright RN (2005) Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In: Proceedings of the 2005 ACM SIGKDD international conference on knowledge discovery and data mining, Chicago, IL, pp 593–599
18. Kantarcıoğlu M, Clifton C (2004) Privacy-preserving distributed mining of association rules on horizontally partitioned data. IEEE Trans Knowl Data Eng 16(9):1026–1037. [Online]. Available: http://csdl.computer.org/comp/trans/tk/2004/09/k1026abs.htm

19. Kargupta H, Datta S, Wang Q, Sivakumar K (2003) On the privacy preserving properties of random data perturbation techniques. In: Proceedings of the third IEEE international conference on data mining (ICDM'03), Melbourne, FL
20. Kargupta H, Datta S, Wang Q, Sivakumar K (2005) Random-data perturbation techniques and privacy-preserving data mining. Knowl Inf Syst 7(4):387–414. [Online]. Available: http://www.springerlink.com/content/va0409rm86aqv9um
21. Karr AF, Lin X, Sanil AP, Reiter JP (2005) Secure regressions on distributed databases. J Comput Graph Stat 14:263–279
22. Lin X, Clifton C, Zhu M (2005) Privacy preserving clustering with distributed EM mixture modeling. Knowl Inf Syst 8(1):68–81
23. Lindell Y, Pinkas B (2002) Privacy preserving data mining. J Cryptol 15(3):177–206
24. Mielikainen T (2004) Privacy problems with anonymized transaction databases. In: Discovery science: 7th international conference proceedings, Lecture notes in computer science, vol 3245, Springer-Verlag, Berlin, January, pp 219–229
25. Naccache D, Stern J (1998) A new public key cryptosystem based on higher residues. In: Proceedings of the 5th ACM conference on computer and communications security, ACM, San Francisco, CA, pp 59–66
26. Okamoto T, Uchiyama S (1998) A new public-key cryptosystem as secure as factoring. In: Advances in cryptology—Eurocrypt '98, Lecture notes in computer science, vol 1403. Springer-Verlag, Berlin, pp 308–318
27. Oliveira S, Zaiane O (2003) Privacy preserving clustering by data transformation. In: Proceedings of the 18th Brazilian symposium on databases, pp 304–318. [Online]. Available: citeseer.ifi.unizh.ch/oliveira03privacy.html
28. Paillier P (1999) Public key cryptosystems based on composite degree residuosity classes. In: Advances in Cryptology—Eurocrypt '99 proceedings, lecture notes in computer science, vol 1592. Springer-Verlag, Berlin, pp 223–238
29. Ravikumar P, Cohen WW, Fienberg SE (2004) A secure protocol for computing string distance metrics. In: Proceedings of the workshop on privacy and security aspects of data mining at the international conference on data mining, pp 40–46
30. Rizvi SJ, Haritsa JR (2002) Maintaining data privacy in association rule mining. In: Proceedings of 28th international conference on very large data bases, VLDB, Hong Kong, pp 682–693. [Online]. Available: http://www.vldb.org/conf/2002/S19P03.pdf
31. Sanil AP, Karr AF, Lin X, Reiter JP (2004) Privacy preserving regression modelling via distributed computation. In: KDD '04: Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining, ACM, New York, pp 677–682
32. Standard for privacy of individually identifiable health information. Fed Regist 66(40), 2001. [Online]. Available: http://www.hhs.gov/ocr/hipaa/finalreg.html
33. Sweeney L, Shamos M (2004) A multiparty computation for randomly ordering players and making random selections. Carnegie Mellon University, School of Computer Science, Tech Rep CMU-ISRI-04-126
34. Vaidya J, Clifton C (2002) Privacy preserving association rule mining in vertically partitioned data. In: The eighth ACM SIGKDD international conference on knowledge discovery and data mining, Edmonton, Alberta, Canada, pp 639–644. [Online]. Available: http://doi.acm.org/10.1145/775047.775142
35. Vaidya J, Clifton C (2003) Privacy-preserving k-means clustering over vertically partitioned data. In: The ninth ACM SIGKDD international conference on knowledge discovery and data mining, Washington, DC, pp 206–215. [Online]. Available: http://doi.acm.org/10.1145/956750.956776
36. Vaidya J, Clifton C (2004) Privacy preserving naïve bayes classifier for vertically partitioned data. In: 2004 SIAM international conference on data mining, Lake Buena Vista, FL, pp 522–526
37. Vaidya J, Clifton C (2004) Privacy-preserving outlier detection. In: Proceedings of the fourth IEEE international conference on data mining (ICDM'04). IEEE Computer Society Press, Los Alamitos, CA, pp 233–240
38. Vaidya J, Clifton C (2005) Privacy-preserving decision trees over vertically partitioned data. In: The 19th annual IFIP WG 11.3 working conference on data and applications security, Storrs, CT, Springer, Berlin Heidelberg New York [Online]. Available: http://dx.doi.org/10.1007/11535706_11

39. Vaidya J, Clifton C (2005) Secure set intersection cardinality with application to association rule mining. J Comput Secur 13(4):593–622
40. Vaidya J, Clifton C, Zhu M (2005) Privacy-preserving data mining, 1st edn., Advances in information security, vol 19, Springer-Verlag, Berlin. [Online]. Available: http://www.springeronline.com/sgw/cda/frontpage/0,11855,4-40356-72-52496494-0,00.html
41. Vapnik VN (1998) Statistical learning theory. Wiley, New York
42. Verykios VS, Bertino E, Fovino IN, Provenza LP, Saygin Y (2004) State-of-the-art in privacy preserving data mining. SIGMOD Rec 33(1):50–57. [Online]. Available: http://www.acm.org/sigmod/record/issues/0403/B1.bertion-sigmod-record2.pdf
43. Wright R, Yang Z (2004) 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
44. Xu S, Zhang J, Han D, Wang J (2006) Singular value decomposition based data distortion strategy for privacy protection. Knowl Inf Syst 10(3):383–397. [Online]. Available: http://www.springerlink.com/content/r5778lt2q3763213
45. Yao AC (1986) How to generate and exchange secrets. In: Proceedings of the 27th IEEE symposium on foundations of computer science, IEEE Press, Los Alamitos, CA, pp 162–167
46. Yu H, Jiang X, Vaidya J (2006) Privacy-preserving SVM using nonlinear kernels on horizontally partitioned data. In: SAC '06: Proceedings of the 2006 ACM symposium on applied computing, ACM, New York, pp 603–610
47. Yu H, Vaidya J (2004) Secure matrix addition. UIOWA Tech Rep UIOWA-CS-04-04. Available: http://hwanjoyu.org/paper/techreport04-04.pdf, Tech. Rep.
48. Yu H, Vaidya J, Jiang X (2006) Privacy-preserving SVM classification on vertically partitioned data. In: Proceedings of PAKDD '06, Lecture notes in computer science, vol 3918. Springer-Verlag, Berlin, pp 647–656. [Online]. Available: http://dx.doi.org/10.1007/11731139_74
49. Yu H, Vaidya J (in press) Privacy preserving linear SVM classification. Submitted for publication to Data & Knowledge Engineering, Elsevier, Science, Amsterdam
50. Zhang N, Wang S, Zhao W (2004) A new scheme on privacy-preserving association rule mining. In: The 8th European conference on principles and practice of knowledge discovery in databases (PKDD 2004), Pisa, Italy. [Online]. Available: http://www.springerlink.com/openurl.asp?genre=article&issn=0302-9743&volume=3202&spage=484
51. Zhu Y, Liu L (2004) Optimal randomization for privacy preserving data mining. In: KDD '04: Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining, ACM, New York, pp 761–766

Author Biographies

Jaideep Vaidya received the Bachelor's degree in Computer Engineering from the University of Mumbai. He received the Master's and the Ph.D. degrees in Computer Science from Purdue University. He is an Assistant Professor in the Management Science and Information Systems Department at Rutgers University. His research interests include data mining and analysis, information security, and privacy. He has received best paper awards for papers in ICDE and SIDKDD. He is a Member of the IEEE Computer Society and the ACM.

Hwanjo Yu received the Ph.D. degree in Computer Science in 2004 from the University of Illinois at Urbana-Champaign. He is an Assistant Professor in the Department of Computer Science at the University of Iowa. His research interests include data mining, machine learning, database, and information systems. He is an Associate Editor of Neurocomputing and served on the NSF Panel in 2006. He has served on the program committees of 2005 ACM SAC on Data Mining track, 2005 and 2006 IEEE ICDM, 2006 ACM CIKM, and 2006 SIAM Data Mining.

Xiaoqian Jiang received the B.S. degree in Computer Science from Shanghai Maritime University, Shanghai, 2003. He received the M.C.S. degree in Computer Science from the University of Iowa, Iowa City, 2005. Currently, he is pursuing a Ph.D. degree from the School of Computer Science, Carnegie Mellon University. His research interests are computer vision, machine learning, data mining, and privacy protection technologies.