Privacy-Preserving Support Vector Machines Learning

Justin Zhan[+], LiWu Chang[*] and Stan Matwin[+]
[+]School of Information Technology & Engineering, University of Ottawa, Canada
[*]Center for High Assurance Computer Systems, Naval Research Laboratory, USA
zhizhan@site.uottawa.ca, lchang@itd.nrl.navy.mil, stan@site.uottawa.ca

Abstract: This paper addresses the problem of data sharing among multiple parties, without disclosing the data between the parties. We focus on sharing of data among parties involved in a data mining task. We study how to share private or confidential data in the following scenario: without disclosing their private data to each other, multiple parties, each having a private data set, want to collaboratively construct support vector machines using a linear, polynomial or sigmoid kernel function. To tackle this problem, we develop a secure protocol for multiple parties to conduct the desired computation. The solution is distributed, i.e., there is no central, trusted party having access to all the data. Instead, we define a protocol using homomorphic encryption techniques to exchange the data while keeping it private. We analyze the protocol in the context of mistakes and malicious attacks, and show its robustness against such attacks. All the parties are treated symmetrically: they all participate in the encryption and in the computation involved in learning support vector machines.

Keywords: Privacy, security, support vector machine, secure multi-party computation.

I. Introduction

In this paper, we address the following problem: multiple parties are cooperating on a data-rich task. Each of the parties owns data pertinent to the aspect of the task addressed by this party. More specifically, the data consists of instances, each party owns her instances but all parties have the same attributes. The overall performance, or even solvability, of this task depends on the ability of performing data mining using all the instances of all the parties. The parties, however, may be unwilling to release their instances to other parties, due to privacy or confidentiality of the data. How can we structure information sharing between the parties so that the data will be shared for the purpose of data mining, while at the same time specific instance values will be kept confidential by the parties to whom they belong? This is the task addressed in this paper. In the privacy-oriented data mining this task is known as data mining with horizontally partitioned data (also known as homogeneous collaboration [15].) Examples of such tasks abound in business, homeland security, coalition building, medical research, etc.

The following scenarios illustrate situations in which this type of collaboration is interesting:(i) Multiple competing supermarkets, each having an extra large set of data records of its customers' buying behaviors, want to conduct data mining on their joint data set for mutual benefit. Since these companies are competitors in the market, they do not want to disclose their customers' information to each other, but they know the results obtained from this collaboration could bring them an advantage over other competitors. (ii) Success of homeland security aiming to counter terrorism depends on combination of strength across different mission areas, effective international collaboration and information sharing to support coalition in which different organizations and nations must share some, but not all, information. Information privacy thus becomes extremely important: all the parties of the collaboration promise to provide their private data to the collaboration, but neither of them wants each other or any other party to learn much about their private data.

Without privacy concerns, all parties can send their data to a trusted central place to conduct the mining. However, in situations with privacy concerns, the parties may not trust anyone. We call this type of problem the *Privacy-preserving Collaborative Data Mining problem*. As stated above, in this paper we are interested in homogeneous collaboration where each party has the same sets of attributes [15] but has different sets of instances.

Data mining includes a number of different tasks, such as association rule mining, classification, and clustering, etc. This paper studies how to learn support vector machines. In the last few years, there has been a surge of interest in Support Vector Machines(SVM)[29, 28]. SVM is a powerful methodology for solving problems in nonlinear classification, function estimation and density estimation which has also led to many other recent developments in kernel based learning methods in gen-

Proceedings of the Fifth International Conference on Electronic Business, Hong Kong, December 5-9, 2005, pp. 477 - 482.

eral [7, 25, 24]. SVMs have been introduced within the context of statistical learning theory and structural risk minimization. As part of the SVM algorithm, one solves convex optimization problems, typically quadratic programs. It has been empirically shown that SVMs have good generalization performance on many applications such as text categorization [13], face detection [20], and handwritten character recognition [16]. Based on the existing SVM learning technologies, we study the problem of learning Support Vector Machines on private data. More precisely, the problem is defined as follows: multiple parties want to build support vector machines on a data set that consists of private data of all the parties, but none of the parties is willing to disclose her raw data to each other or any other parties. We develop a secure protocol, based on homomorphic cryptography and random perturbation techniques, to tackle the problem. An important feature of our approach is its distributed character, i.e. there is no single, centralized authority that all parties need to trust. Instead, the computation is distributed among parties, and its structure and the use of homomorphic encryption ensures privacy of the data.

The paper is organized as follows: The related work is discussed in Section 2. We describe the SVMs training procedure in Section 3. We then present our proposed secure protocols in Section 4. We give our conclusion in Section 5.

II. Related Work

II.1 Secure Multi-Party Computation

A Secure Multi-party Computation (SMC) problem deals with computing any function on any input, in a distributed network where each participant holds one of the inputs, while ensuring that no more information is revealed to a participant in the computation than can be inferred from that participant's input and output. The SMC problem literature was introduced by Yao [31]. It has been proved that for any polynomial function, there is a secure multi-party computation solution [12]. The approach used is as follows: the function F to be computed is firstly represented as a combinatorial circuit, and then the parties run a short protocol for every gate in the circuit. Every participant gets corresponding shares of the input wires and the output wires for every gate. This approach, though appealing in its generality and simplicity, is highly impractical for large datasets.

II.2 Privacy-Preserving Data Mining

In early work on privacy-preserving data mining, Lindell and Pinkas [17] propose a solution to privacy-preserving classification problem using oblivious transfer protocol, a powerful tool developed by secure multi-party computation (SMC) research. The techniques based on SMC for efficiently dealing with large data sets have been addressed in [14], where a solution to the association rule mining problem for the case of two parties was proposed.

Randomization approaches were firstly proposed by Agrawal and Srikant in [3] to solve privacy-preserving data mining problem. In addition to perturbation, aggregation of data values [26] provides another alternative to mask the actual data values. In [1], authors studied the problem of computing the kth-ranked element. Dwork and Nissim [9] showed how to learn certain types of boolean functions from statistical databases in terms of a measure of probability difference with respect to probabilistic implication, where data are perturbed with noise for the release of statistics. In this paper, we focus on privacy-preserving among the inter-party computation.

Homomorphic encryption [21], which transforms multiplication of encrypted plaintexts into the encryption of the sum of the plaintexts, has recently been used in secure multi-party computation. For instance, Freedmen, Nissim and Pinkas [10] applied it for set intersection. For computing set intersection, unlike [10], [2] and [27] proposed an approach based on commutative encryption. The work most related to ours is [30], where Wright and Yang applied homomorphic encryption [21] to the Bayesian networks induction for the case of *two* parties. The work that are closely related ours is [32], where Zhan et.al. present secure protocols for learning support vector machine over vertically partitioned data. In this paper, we develop a secure protocol, based on homomorphic encryption and random perturbation techniques, for multiple parties to build SVMs over horizontally partitioned data without compromising their data privacy.

III. Learning SVMs On Private Data

Support vector machines were invented by Vapnik [29] in 1982. The idea consists of mapping the space of input examples into a high-dimensional feature space, so that the optimal separating hyperplane built on this space allow a good generalization capacity. The input examples become linearly or almost linearly separable in the high dimensional space through selecting an adequate mapping [28]. Research on SVMs is extensive since it was invented. However, to our best knowledge, there is no effort on learning SVMs on private data. In this paper, our goal is to provide a privacy-preserving algorithm for multi-parties to collaboratively learn SVMs without compromising their data privacy.

III.1 Problem

We consider the scenario where multiple parties P_1, P_2, $\cdots$, and P_n, each having a private data set (denoted by D_1, D_2, $\cdots$ and D_n respectively), want to collaboratively learn SVMs on the concatenation of their data sets. Because they are concerned about their data privacy, neither party is willing to disclose its actual data set to others. Specially, we consider the homogeneous collaboration where each data set contains the same number of attributes but different set of instances. Let m be the total number of attributes in each data set. Let N be the total number of instances, N_1 is the number of instances for P_1, N_2 is the number of instances for P_2, $\cdots$, and N_n is the number of instances for P_n. We further assume that the class labels are shared but the instance identifiers and actual attribute values are kept confidential.

III.2 Overview of Support Vector Machine

SVM is primarily a two-class classifier for which the optimization criterion is the width of the margin between the different classes. In the linear form, the formula for output of a SVM is

$$u = \overrightarrow{w} \cdot \overrightarrow{x} + b, \tag{1}$$

where $\overrightarrow{w}$ is the normal vector to the hyperplane and $\overrightarrow{x}$ is the input vector. To maximize margin, we need minimize the following [5]:

$$\min_{\mathrm{w,b}} \frac{1}{2}||\overrightarrow{w}||^2, \tag{2}$$

subject to $y_i(\overrightarrow{w} \cdot \overrightarrow{x_i} + b) \geq 1, \forall i$, where $\overrightarrow{x_i}$ is the ith training example, and y_i is the correct output of the SVM for the ith training example. The value y_i is $+1$ (resp. -1) for the positive (resp. negative) examples in a class.

Through introducing Lagrangian multipliers, the above optimization can be converted into a dual quadratic optimization problem.

$$\min_{\overrightarrow{\alpha}} \Psi(\overrightarrow{\alpha}) = \min_{\alpha_\mathrm{i},\alpha_\mathrm{j}} \frac{1}{2} \sum_{\mathrm{i,j=1}}^{\mathrm{N}} \alpha_\mathrm{i}\alpha_\mathrm{j}\mathrm{y_i}\mathrm{y_j}\mathrm{K}(\overrightarrow{x_i}, \overrightarrow{x_j}) - \sum_{\mathrm{i=1}}^{\mathrm{N}} \alpha_\mathrm{i}, \tag{3}$$

where α_i are the Lagrange multipliers, $\overrightarrow{\alpha} = \alpha_1, \alpha_2, \cdots, \alpha_N$, subject to inequality constraints: $\alpha_i \geq 0, \forall i$, and linear equality constraint: $\sum_{i=1}^{N} y_i\alpha_i = 0$.

By solving the dual optimization problem, one obtains the coefficients α_i, $i = 1, \cdots, N$, from which the normal vector $\overrightarrow{w}$ and the threshold b can be derived [22].

To deal with non-linearly separable data in feature space, Cortes and Vapnik [6] introduced slack-variables to relax the hard-margin constraints. The modification is:

$$\min \frac{1}{2}||\overrightarrow{w}||^2 + \mathrm{C}\sum_{\mathrm{i=1}}^{\mathrm{N}} \xi_\mathrm{i} \tag{4}$$

subject to $y_i(\overrightarrow{w} \cdot \overrightarrow{x_i} + b) \geq 1 - \xi_i, \forall i$, where ξ_i is slack variable that allows margin failure and constant $C > 0$ determines the trade-off between the empirical error and the complexity term. This leads to dual quadratic problem involving Eq.[3] subject to the constraints $C \geq \alpha_i \geq 0, \forall i$ and $\sum_{i=1}^{N} y_i\alpha_i = 0$.

To solve the dual quadratic problem, we apply sequential minimal optimization [22] which is a very efficient algorithm for training SVMs.

III.3 Sequential Minimal Optimization

Sequential Minimal Optimization (SMO) [22] is a simple algorithm that can efficiently solve the SVM quadratic optimization (QO) problem. Instead of directly tackle the QO problem, it decomposes the overall QO problem into QO sub-problems based on Osunna's convergence theorem [20]. At each step, SMO chooses two Lagarange multipliers to jointly optimize, find the optimal values for these multipliers, and updates the SVM to reflect the new optimal values.

In order to solve for the two Lagrange multipliers, SMO firstly computes the constraints on these multipliers and then solves for the constrained minimum. Normally, the objective function is positive definite, SMO computes the minimum along the direction of the linear constraints $\sum_{i=1}^{2} y_i\alpha_i = 0$ within the boundary $C \geq \alpha_i \geq 0,\ i = 1, 2.$

$$\alpha_2^{\mathrm{new}} = \alpha_2 + \mathrm{y_2}(\mathrm{E_1} - \mathrm{E_2})\,\eta, \tag{5}$$

where $E_i = y_i\alpha_i K(\overrightarrow{x_i}, \overrightarrow{x}) - y_i$ is the error on the ith training example, $\overrightarrow{x_i}$ is the stored training vector and $\overrightarrow{x}$ is the input vector, and η is the second derivative of Eq.[3] along the direction of the above linear constraints:

$$\eta = K(\overrightarrow{x_1}, \overrightarrow{x_1}) + K(\overrightarrow{x_2}, \overrightarrow{x_2}) - 2K(\overrightarrow{x_1}, \overrightarrow{x_2}). \tag{6}$$

Next step, the constrained minimum is found by clipping the unconstrained minimum to the ends of the line segment: $\alpha_2^{new,clipped}$ is equal to H if $\alpha_2^{new} \geq H$, is equal to α_2^{new} if $L < \alpha_2^{new} < H$, and is equal to $\alpha_2^{new,clipped} = L$ if $\alpha_2^{new} \leq L$. If the target y_1 is not equal to the target y_2, $L = \max(0, \alpha_2 - \alpha_1)$, $H = \min(C, C + \alpha_2 - \alpha_1)$. If the target y_1 is equal to the target y_2, $L = \max(0, \alpha_2 + \alpha_1 - C)$, $H = \min(C, \alpha_2 + \alpha_1)$.

The value of α_1 is computed from the new, clipped, α_2:

$$\alpha_1^{\mathrm{new}} = \alpha_1 + \mathrm{s}(\alpha_2 - \alpha_2^{\mathrm{new,clipped}}), \tag{7}$$

where $s = y_1 y_2$.

In the procedure of sequential minimal optimization, the only step accessing the actual attribute values is the computation of the kernel function K. Kernel functions have various forms. Three types of kernel functions are considered: they are the linear kernel function $K = (\overrightarrow{a} \cdot \overrightarrow{b})$, the polynomial kernel function $K = ((\overrightarrow{a} \cdot \overrightarrow{b}) + \theta)^d$, where $d \in N, \theta \in \mathcal{R}$ are constants, and the sigmoid kernel function $K = \tanh((\kappa(\overrightarrow{a} \cdot \overrightarrow{b})) + \theta)$, where $\kappa, \theta \in \mathcal{R}$ are constants, for instances $\overrightarrow{a}$ and $\overrightarrow{b}$.

To compute these types of kernel functions, one needs to compute the inner product between two instances. If the two instances belong to the same party, this party can compute the inner product by herself; if one instance (e.g., $\overrightarrow{x_1}$) belongs to one party (e.g., P_1), and the other instance (e.g., $\overrightarrow{x_2}$) belongs to another party (e.g., P_2), then P_1 can compute $(\overrightarrow{x_1} \cdot \overrightarrow{x_1})$ and P_2 can compute $(\overrightarrow{x_2} \cdot \overrightarrow{x_2})$. However, to compute $(\overrightarrow{x_1} \cdot \overrightarrow{x_2})$, different parties have to collaborate. How to conduct this inner product computation across parties without compromising each party's data privacy presents a great challenge. In next section, secure protocols are developed to tackle this challenge.

IV. Protocols

IV.1 Introducing Homomorphic Encryption

The concept of homomorphic encryption was originally proposed in [23]. Since then, many such systems have been proposed [4, 18, 19, 21]. We observe that some homomorphic encryption schemes, such as [8], are not robust against chosen plaintext attacks. However, we base our secure protocols on [21], which is semantically secure [11].

In our secure protocols, we use additive homomorphism offered by [21]. In particular, we utilize the following characterizer of the homomorphic encryption functions: $e(a_1) \times e(a_2) = e(a_1 + a_2)$ where e is an encryption function; a_1 and a_2 are the data to be encrypted. Because of the property of associativity, $e(a_1 + a_2 + .. + a_n)$ can be computed as $e(a_1) \times e(a_2) \times \cdots \times e(a_n)$ where $e(a_i) \neq 0$. That is

$$d(e(a_1 + a_2 + \cdots + a_n)) = d(e(a_1) \times e(a_2) \times \cdots \times e(a_n)) \quad (8)$$

$$d(e(a_1)^{a_2}) = d(e(a_1 a_2)) \quad (9)$$

IV.2 A Secure Protocol

Let's assume that P_1 has an instance vector $\overrightarrow{x_1}$ and P_2 has an instance vector $\overrightarrow{x_2}$. Both vectors have m elements. We use x_{1i} to denote the ith element in vector $\overrightarrow{x_1}$, and x_{2i} to denote the ith element in vector $\overrightarrow{x_2}$. In order to compute the $K(\overrightarrow{x_1}, \overrightarrow{x_2})$, the key issue is how P_1 and P_2 compute the inner product between $\overrightarrow{x_1}$ and $\overrightarrow{x_2}$ without disclosing them to each other. In our secure protocol, P_1 adds a random number to each of her actual data values, encrypts the masked values, and sends the encrypted masked terms to P_2. By adding the random numbers, P_2 is prevented from guessing P_1's actual values based on encryption patterns. Firstly, one of parties is randomly chosen as a key generator. For simplicity, let's assume P_1 is selected as the key generator. P_1 generates a cryptographic key pair (d, e) of a semantically-secure homomorphic encryption scheme and publishes its public key e. P_1 applies the encryption key to each element of x_1 (e.g., $e(x_{1i} + r_i)$). P_2 computes $e(\overrightarrow{x_1} \cdot \overrightarrow{x_2})$. He then sends $e(\overrightarrow{x_1} \cdot \overrightarrow{x_2})$ to P_1 who decrypts it and gets $(\overrightarrow{x_1} \cdot \overrightarrow{x_2})$.

We describe this more formally as

Protocol 1 *INPUT: P_1's input is a vector $\overrightarrow{x_1} = \{x_{11}, x_{12}, \cdots, x_{1m}\}$, and P_2's input is a vector $\overrightarrow{x_2} = \{x_{21}, x_{22}, \cdots, x_{2m}\}$. The elements in the input vectors are taken from the real number domain.*

1. P_1 performs the following operations:
   (a) She computes $e(x_{1i} + r_i)$s ($i \in [1, m]$) and sends them to P_2. r_i, known only by P_1, is a random number in real domain.
   (b) She computes $e(-r_i)$s ($i \in [1, m]$) and sends them to P_2.
2. P_2 performs the following operations:
   (a) He computes $t_1 = e(x_{11} + r_1)^{x_{21}} = e(x_{11} \cdot x_{21} + r_1 x_{21})$, $t_2 = e(x_{12} + r_2)^{x_{22}} = e(x_{12} \cdot x_{22} + r_2 x_{22})$, $\cdots$, $t_m = e(x_{1m})^{x_{2m}} = e(x_{1m} \cdot x_{2m} + r_m x_{2m})$.
   (b) He computes $t_1 \times t_2 \times \cdots \times t_m = e(x_{11} \cdot x_{21} + x_{12} \cdot x_{22} + \cdots + x_{1m} \cdot x_{2m} + r_1 x_{21} + r_2 x_{22} + \cdots + r_m x_{2m}) = e(\overrightarrow{x_1} \cdot \overrightarrow{x_2} + \sum_{i=1}^{m} r_i x_{2i})$.
   (c) He computes $e(-r_i)^{x_{2i}} = e(-r_i x_{2i})$ for $i \in [1, m]$.
   (d) He computes $e(\overrightarrow{x_1} \cdot \overrightarrow{x_2} + \sum_{i=1}^{m} r_i x_{2i}) \times e(-r_1 x_{21}) \times e(-r_2 x_{22}) \times \cdots \times e(-r_m x_{2m}) = e(\overrightarrow{x_1} \cdot \overrightarrow{x_2})$.

We need to show that the above protocol is correct, and that it preserves the privacy of P_1 and P_2 as postulated in Sec. 3.1.

Lemma 1 *(Correctness). Protocol 1 correctly computes the inner product $(\overrightarrow{x_1} \cdot \overrightarrow{x_2})$ against semi-honest parties.*

Proof When P_2 receives each encrypted element $e(x_{1i} + r_i)$ and $e(-r_i)$, he computes $\sum_{i=1}^{m} e(x_{1i} + r_i)^{x_{2i}}$ which, according to Eq.(9), is equal to $e(\sum_{i=1}^{m} x_{1i} \cdot x_{2i} + \sum_{i=1}^{m} r_i x_{2i})$ for all $i \in [1, m]$. He then computes $e(\overrightarrow{x_1} \cdot \overrightarrow{x_2} +$

$\sum_{i=1}^{m} r_i x_{2i}) \times e(-r_1 x_{21}) \times e(-r_2 x_{22}) \times \cdots \times e(-r_m x_{2m}) = e(\overrightarrow{x_1} \cdot \overrightarrow{x_2})$ according to Eq.(8). After that, he sends it to P_1 who computes $d(e(\overrightarrow{x_1} \cdot \overrightarrow{x_2})) = (\overrightarrow{x_1} \cdot \overrightarrow{x_2})$. Therefore, $(\overrightarrow{x_1} \cdot \overrightarrow{x_2})$ is correctly computed.

∎

Lemma 2 *(Privacy-Preserving). Assuming the parties follow the protocol, the private data are securely protected.*

Proof There are 2 points we need analyze. (1) Whether P_1 can obtain P_2's private data. There is no information that P_2 sends to P_1, thus, P_2's private data cannot be disclosed to P_1. (2) Whether P_2 can obtain P_1's private data. What P_2 receives from P_1 is encrypted and masked element of P_1's data. Since P_2 has no decryption key and doesn't know the random number used by P_1, it is impossible that P_2 can obtain P_1's private data. ∎

Lemma 3 *(Efficiency). Protocol 1 is efficient from both computation and communication point of view.*

Proof To prove the efficiency, we need conduct complexity analysis of the protocol. The bit-wise communication cost of this protocol is $(2m + 1)\alpha$ where α is the number of bits for each transmitted element. The following contributes to the computational cost: (1) 2m encryptions; (2) 2m exponentiations; (2) 2m-1 multiplications. Therefore, the protocol is sufficient fast.

∎

V. Conclusion

In this paper, we consider the problem of collaboratively learning Support Vector Machines on private data. We develop a secure collaborative protocol based on semantically secure homomorphic encryption scheme. In our protocol, the parties do not need to send all their data to a central, trusted party. Instead, we use the homomorphic encryption and random perturbation techniques to conduct the computations across the parties without compromising their data privacy. As future work, we will develop secure protocols for the cases where more kernel functions are applied. We will also apply our technique to other data mining computations, such as secure collaborative clustering.

References

[1] G. Aggarwal, N. Mishra, and B. Pinkas. Secure computation of the k th-ranked element. In *EUROCRYPT pp 40-55*, 2004.

[2] R. Agrawal, A. Evfimievski, , and R. Srikant. Information sharing across private databases. In *Proceedings of ACM SIGMOD ICMD, pp 86-97*, 2003.

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

[4] J. Benaloh. Dense probabilistic encryption. In *Proceedings of the Workshop on Selected Areas of Cryptography, pp. 120-128, Kingston, Ontario, May*, 1994.

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

[6] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[7] Shawe-Taylor J. Cristianini N. An introduction to support vector machines. In *Cambridge University Press.*

[8] J. Domingo-Ferrer. A provably secure additive and multiplicative privacy homomorphism. In *Information Security Conference, 471-483, 2002.*

[9] C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases.

[10] M. Freedman, K. Nissim, and B. Pinkas. Effiicent private matching and set intersection. In *EUROCRYPT pp 1-19*, 2004.

[11] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikainen. On secure scalar product computation for privacy-preserving data mining. In *Proceedings of The 7th Annual International Conference in Information Security and Cryptology (ICISC 2004), volume 3506 of Lecture Notes in Computer Science, pages 104–120, Seoul, Korea, December 2–3, 2004, Springer-Verlag*, 2004.

[12] O. Goldreich. Secure multi-party computation (working draft). http://www.wisdom.weizmann.ac.il /home/oded/public_html/foc.html, 1998.

[13] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[14] J.Vaidya and C.W.Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada.*

[15] M. Kantarcioglu and C. Clifton. Privacy preserving data mining of association rules on horizontally partitioned data. In *Transactions on Knowledge and Data Engineering, IEEE Computer Society Press, Los Alamitos, CA.*

[16] Y. LeCun, L. Botou, L. Jackel, H. Drucker, C. Cortes, J. Denker, I. Guyon, U. Muller, E. Sackinger, P. Simard, and V. Vapnik. Learning algorithms for classification: A comparison on handwritten digit recognition, 1995.

[17] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Advances in Cryptology - Crypto2000, Lecture Notes in Computer Science*, volume 1880, 2000.

[18] D. Naccache and J. Stern. A new public key cryptosystem based on higher residues. In *Proceedings of the 5th ACM conference on Computer and Communication Security, pp. 59-66, San Francisco, California, United States*, 1998.

[19] T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In *Eurocrypt'98, LNCS 1403, pp.308-318*, 1998.

[20] Freund-R. Girosi F. Osuna, E. Training support vector machines: An application to face detection. In *Proceedings of Computer Vision and Pattern Recognition, 130-136.*

[21] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptography - EUROCRYPT '99, pp 223-238, Prague, Czech Republic*, May 1999.

[22] J. Platt. Sequetial minimal optimization: A fast algorithm for training support vector machines. In *Technical Report MST-TR-98-14. Microsoft Research, 1998.*

[23] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation, eds. R. A. DeMillo et al., Academic Press, pp. 169-179.*, 1978.

[24] Mller K.-R. Schlkopf B., Smola A. J. Nonlinear component analysis as a kernel eigenvalue problem. In *Neural Computation, 10, 1299-1319.*

[25] Smola A. (Eds.) Schlkopf B., Burges C. Advances in kernel methods - support vector learning. In *MIT Press.*

[26] L. Sweeney. k-anonymity: a model for protecting privacy. In *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems 10 (5), pp 557–570*, 2002.

[27] J. Vaidya and C. Clifton. Secure set intersection cardinality with application to association rule mining. In *Journal of Computer Security, IOS Press, to appear.*

[28] V. Vapnik. The nature of statistical learning theory. In *Springer-Verlag, New York, 1995.*

[29] V. N. Vapnik. Estimation of dependences based on empirical data. In *Springer-Verlag, New York, 1982. 22.*

[30] 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 (KDD)*, 2004.

[31] A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, 1982.

[32] Z. Zhan, S. Matwin, and L. Chang. Building support vector machines on private data. In *International Conference on Artificial Intelligence, to appear*, 2005.