Towards Privacy-Preserving Model Selection*

Zhiqiang Yang[1], Sheng Zhong[2], and Rebecca N. Wright[3]

[1] Department of Computer Science, Stevens Institute of Technology, Hoboken, NJ, 07030, USA, zyang@cs.stevens.edu
[2] Department of Computer Science, SUNY Buffalo, Buffalo, NY, 14260, USA, szhong@cse.buffalo.edu
[3] Department of Computer Science and DIMACS, Rutgers University, Piscataway, NJ, 08854, USA, rebecca.wright@rutgers.edu

Abstract. Model selection is an important problem in statistics, machine learning, and data mining. In this paper, we investigate the problem of enabling multiple parties to perform model selection on their distributed data in a privacy-preserving fashion without revealing their data to each other. We specifically study cross validation, a standard method of model selection, in the setting in which two parties hold a vertically partitioned database. For a specific kind of vertical partitioning, we show how the participants can carry out privacy-preserving cross validation in order to select among a number of candidate models without revealing their data to each other.

1 Introduction

In today's world, a staggering amount of data, much of it sensitive, is distributed among a variety of data owners, collectors, and aggregators. Data mining provides the power to extract useful knowledge from this data. However, privacy concerns may prevent different parties from sharing their data with others. A major challenge is how to realize the utility of this distributed data while also protecting data privacy.

Privacy-preserving data mining provides data mining algorithms in which the goal is to compute or approximate the output of one or more particular algorithms applied to the joint data, without revealing anything else, or at least anything else sensitive, about the data.

To date, work on distributed privacy-preserving data mining has been primarily limited to providing privacy-preserving versions of particular data mining algorithms. However, the data miner's task rarely starts and ends with running a particular data mining algorithm. In particular, a data miner seeking to model some data will often run a number of different kinds of data mining algorithms and then perform some kind of model selection to determine which of the resulting models to use. If privacy-preserving methods are used for determining many

* This work was supported in part by the National Science Foundation under Grant No. CCR-0331584 and by the Department of Homeland Security under ONR Grant N00014-07-1-0159.

In *Proceedings of the First SIGKDD International Workshop on Privacy, Security, and Trust in KDD (PinKDD'07), LNCS 4890, Springer, 2008.*

models, but then the model selection either is carried out without maintaining privacy or cannot be carried out due to privacy constraints, then the desired privacy and utility cannot simultaneously be achieved.

In this paper, we introduce the notion of privacy-preserving model selection. We specifically consider cross validation, which is a popular method for model selection. In cross validation, a number of different models are generated on a portion of the data. It is then determined how well the resulting models perform on the rest of the data, and the highest performing model is selected. Cross validation can also be used to validate a single model learned from training data on test data not used in the generation of the model, to determine whether the model performs sufficiently well and limit the possibility of choosing a model that overfits the data.

We provide a partial solution to privacy-preserving model selection via cross validation. We assume a very specific kind of vertical partitioning, which has previously been considered by Vaidya and Clifton [31], in which one party holds all the data except the class labels, and a second party holds all the class labels. In this setting, we show how to perform model selection using cross validation in a privacy-preserving manner, without revealing the parties' data to each other.

A practical example of this kind of partitioning might occur in a research project seeking to explore the relationship between certain criminal activities and the medical histories of people involved in these activities. A local hospital has a database of medical histories, while the police department has the criminal records. Both the hospital and police department would like to provide assistance to this project, but neither of them is willing or legally able to reveal their data to the other. It is therefore a technical challenge to find the right model over this distributed database in a privacy-preserving manner. Specifically, we can view the medical histories and the criminal records as two parts of a vertically partitioned database. We simplify the criminal records to labels on local residents for whether they are involved in the criminal activities. Then the question becomes finding the right model to predict this label on an individual using his or her medical history data. We require that the medical histories not be revealed to the police department and that the labels not be revealed to the hospital.

Our main contribution is a privacy-preserving model selection protocol for this setting. Specifically, there is a database vertically partitioned between two participants. One participant has all the data except the class labels; the other participant has all the class labels. Our privacy-preserving cross validation solution enables the parties to privately determine the best among a number of candidate models for the data, thereby extending the privacy of the data from the initial model computation through to the model selection step.

We begin by discussing related work in Section 2. We introduce some cryptographic primitives in Sections 3 and 4. Our main protocol is shown in Section 5. In Section 6, we discuss possible extensions including generalizing our solution to arbitrary vertically partitioned data and determining which of a number of models is best without revealing the models that are not chosen.

2 Related Work

Existing techniques in privacy-preserving data mining can largely be categorized into two approaches. One approach adopts cryptographic techniques to provide secure solutions in distributed settings, as pioneered by Lindell and Pinkas [25]. Another approach randomizes the original data such that certain underlying patterns are still kept in the randomized data, as pioneered by Agrawal and Srikant [3]. Generally, the cryptographic approach can provide solutions with perfect accuracy and perfect privacy. The randomization approach is much more efficient than the cryptographic approach, but typically suffers a tradeoff between privacy and accuracy.

In the randomization approach, original data is randomized by adding noise so that the original data is disguised but patterns of interest persist. The randomization approach enables learning data mining models from the disguised data. Different randomization approaches have been proposed to learn different data mining models, such as decision trees [3,8] and association rules [10,11,28]. Several privacy definitions for the randomization setting have been proposed to achieve different levels of privacy protection [1, 3, 10], though privacy problems with randomization approach have also been discussed [17, 23].

In the cryptographic approach, which we follow in this paper, the goal is to enable distributed learning of data mining models across different databases without the database owners revealing anything about their data to each other beyond what can be inferred from the final result. In principle, general-purpose secure multiparty computation protocols [16, 35] can provide solutions to any distributed privacy-preserving data mining problem.

However, these general-purpose protocols are typically not efficient enough for use in practice when the input data is very large. Therefore, more efficient special-purpose privacy-preserving protocols have been proposed for many special cases. These address a number of different learning problems across distributed databases, such as association rule mining [21,29], ID3 trees [25], clustering [19,30], naive Bayes classification [22,31], and Bayesian networks [27,34], as well as a variety of privacy-preserving primitives for simple statistical computations including scalar product [4,7,13,14,29,33], finding common elements [2,13], and computing correlation matrices [26].

In the cryptographic approach, privacy is quantified using variants of the standard cryptographic definition for secure multiparty computation. Intuitively, parties involved in the privacy-preserving distributed protocols should learn only the data mining results that are their intended outputs, and nothing else.

Most privacy-preserving data mining solutions to date address typical data mining algorithms, but do not address necessary preprocessing and postprocessing steps. Recent work addresses privacy preservation during the preprocessing step [20]. In this work, we begin the exploration of extending the preservation of privacy to the postprocessing step, thereby maintaining privacy throughout the data mining process.

3 Cryptographic Tools

In this section, we briefly overview cryptographic concepts and primitives that we use.

3.1 Privacy Definition

In this paper, we define privacy using a standard definition used in secure multiparty computation [15]. In particular, we consider the privacy definition in the model of semi-honest adversaries. A semi-honest adversary is assumed to follow its specified instructions, but will try to gain as much information as possible about other parties' inputs from the messages it receives.

The proofs of privacy in this paper are carried out using the simulation paradigm [15]. Formally, let Π be a 2-party protocol for computing a function $f : (x_1, x_2) \rightarrow (y_1, y_2)$. The *view* of the ith party ($i \in \{1, 2\}$) during an execution of Π, denoted by $\mathsf{view}_i(x_1, x_2)$, consists of the ith party's input x_i, all messages received by the ith party, and all internal coin flips of the ith party. We say that *Π privately computes f against semi-honest adversaries* if, for each i, there exists a probabilistic polynomial-time algorithm S_i (which is called a *simulator*), such that

$$\{S_i(x_i, y_i)\}_{x_1,x_2} \stackrel{\mathsf{c}}{\equiv} \{(\mathsf{view}_i(x_1, x_2)\}_{x_1,x_2},$$

where $\stackrel{\mathsf{c}}{\equiv}$ denotes *computational indistinguishability*. (See [15] for the definition of computational indistinguishability. Intuitively, it states that a polynomially-bounded computation cannot distinguish between the two distributions given samples from them.)

Intuitively, this definition states that, based on the input and output of each participant, we should be able to "simulate" the view of that participant. Therefore, each participant learns nothing during the computation that would not be learned if Alice and Bob gave their data to a trusted third party who computed the results y_1 and y_2 and returned them to Alice and Bob, respectively.

As a privacy definition, this definition has some advantages but it also has some limitations. Among its advantages are that it allows provable guarantees that nothing was leaked during the computation, and that if multiple subprotocols are combined properly, their combination does not leak any information. A notable limitation of this definition is that it does not say anything about the privacy of the final result, leaving that determination as a separate privacy decision.

3.2 ElGamal Cryptosystem

A public key cryptosystem consists of three algorithms: the *key generation algorithm*, the *encryption algorithm*, and the *decryption algorithm*. We make use of the ElGamal cryptosystem [9].

In the ElGamal cryptosystem, the key generation algorithm generates the *parameters* (G, q, g, x), where G is a cyclic group of order q with generator g, and x is randomly chosen from $\{0, \ldots, q-1\}$. The *public key* is (h, G, q, g) where $h = g^x$, and the *private key* is x.

In order to encrypt a message m to Alice under her public key (h, G, q, g), Bob computes $(c_1 = m \cdot h^k, c_2 = g^k)$, where k is randomly chosen from $\{0, \ldots, q-1\}$. To decrypt a ciphertext (c_1, c_2) with the private key x, Alice computes $c_1(c_2^x)^{-1}$ as the plaintext message.

ElGamal encryption is *semantically secure* under the Decisional Diffie-Hellman (DDH) assumption [5], which we assume throughout this paper. One group family in which DDH is commonly assumed to be intractable is the quadratic residue subgroup of $\mathbb{Z}_p^*$ (the multiplicative group mod p) where p is a *safe* prime (i.e., a prime number of the form $p = 2p' + 1$ for a prime p'). ElGamal encryption has a useful *randomization property*. Specifically, given an encryption of M, it is possible to compute a different (and random) encryption of M without knowing the private key.

4 Privacy-Preserving Hamming Distance and Generalized Hamming Distance

In this section, we provide new, simple, efficient, privacy-preserving protocols for computing the Hamming distance and generalized Hamming distance between two vectors. These will be used in our main protocol.

4.1 Privacy-Preserving Hamming Distance

In this protocol, Alice has a vector $A = (a_1, ..., a_n)$ and Bob has a vector $B = (b_1, ..., b_n)$, where A and B contain only binary values. In our setting, Alice is supposed to learn the Hamming distance of their two vectors, and Alice and Bob are supposed to learn nothing else about each other's vectors. (In the semi-honest setting, such a protocol can easily be transformed into a protocol where both Alice and Bob learn the result, by having Alice tell Bob the answer.)

We note that private solutions already exist for this problem. For example, an efficient solution is given by Jagannathan and Wright [20] based on homomorphic encryption. Yao's secure two-party computation could be used [35]; it computes the result based on computation using a "garbled" circuit. Alternately, the secure two-party computation techniques of Boneh, Goh, and Nissim [6] could be used, in the form where the output is multiple bits; this relies on computationally expensive bilinear pairing, as well as on a new computational assumption. We also note that if one is willing to accept an approximation to the Hamming distance, it is possible to achieve this with sublinear communication complexity while meeting the privacy requirements [12, 18].

In this section, we describe a simple, efficient, alternative solution based on the ElGamal cryptosystem. As shown in the following section, a modification of this solution also solves the generalized Hamming distance problem.

Privacy-Preserving Hamming Distance Protocol

Input: Vectors $A = (a_1, \ldots, a_n)$ and $B = (b_1, \ldots, b_n)$ held by Alice and Bob, respectively, such that $a_i, b_i \in \{0, 1\}$ for $1 \leq i \leq n$.

Output: Alice learns the Hamming distance of A and B.

1. For $1 \leq i \leq n$, if $a_i = 0$, then Alice sends $e_i = E(g)$ to Bob; Otherwise, Alice sends $e_i = E(g^{-1})$ to Bob. (Obviously, fresh, independent randomness is used in generating each of these encryptions.) For each i, the resulting encryption is a two-part ciphertext $e_i = (c_{i,1}, c_{i,2})$.
2. For $1 \leq i \leq n$, if $b_i = 0$, then Bob rerandomizes e_i to get e'_i; otherwise, Bob sets e'_i to a rerandomization of $e_i^{-1} = ((c_{i,1})^{-1}, (c_{i,2})^{-1})$. Bob sends a permuted vector including all e'_i to Alice.
3. Alice decrypts all received e'_i and counts how many of the decryptions are equal to g^{-1}. This number is equal to $\sum_{i=1}^{n}(a_i \oplus b_i) = \mathsf{dist}(A, B)$.

Fig. 1. Privacy-Preserving Hamming Distance Protocol

In our protocol, we assume Alice has an ElGamal key pair (x, y) ($x \in [0, q-1]$, where q is the order of G; $y \in G$) such that $y = g^x \in G$. Here, x is the private key, which is known only to Alice, and y is the public key, which is also known to Bob. We use $E(m)$ to denote an encryption of m by public key y. All computations in the protocol and throughout this paper take place in G, which is chosen large enough to ensure that the final distance result is correct as an integer. The output of this protocol is the Hamming distance $\mathsf{dist}(A, B) = \sum_{i=1}^{n}(a_i \oplus b_i)$. The basic idea is that we use g to represent 0 and g^{-1} to represent 1. The protocol is shown in Figure 1.

Theorem 1. *Under the DDH assumption, the protocol in Figure 1 for binary-valued inputs privately computes the Hamming distance in the semi-honest model.*

Proof. We first show correctness—i.e., that Alice's output is the correct Hamming distance. In Step 1, for $1 \leq i \leq n$, Alice computes

$$e_i = (c_{i,1}, c_{i,2}) = \begin{cases} E(g) & \text{if } a_i = 0 \\ E(g^{-1}) & \text{if } a_i = 1 \end{cases}$$

and sends these to Bob. In Step 2, for $1 \leq i \leq n$, Bob produces e'_i. If $b_i = 0$, then e'_i is a rerandomization of the encryption e_i. In this case, e'_i encrypts the same cleartext g or g^{-1} that e_i does (even though Bob does not himself know this cleartext). If, on the other hand, $b_i = 1$, then Bob sets e'_i to be a rerandomization of $({c_{i,1}}^{-1}, {c_{i,2}}^{-1})$. Assuming k_i is the random value used in Alice's encryption of $m_i \in \{g, g^{-1}\}$, then:

$$({c_{i,1}}^{-1}, {c_{i,2}}^{-1}) = ((m_i \cdot h^{k_i})^{-1}, (g^{k_i})^{-1}) = ({m_i}^{-1} \cdot h^{-k_i}, g^{-k_i}).$$

Once rerandomized so as to use fresh randomization, this is a valid encryption of ${m_i}^{-1}$. Thus, if $m_i = g$, then e'_i is an encryption of g^{-1}, and if $m_i = g^{-1}$, then

e'_i is an encryption of g. It further follows that e'_i is an encryption of g^{-1} if and only if $a_i \neq b_i$. (If $a_i = b_i$, then e'_i is an encryption of g.) Therefore, the number of g^{-1} decryptions Alice obtains in Step 3 is the desired Hamming distance.

To show privacy, we need to demonstrate simulators S_1 for Alice and S_2 for Bob. We construct S_1 as follows. S_1 simulates all internal coin flips of Alice as described in the protocol. S_1 simulates the message from Bob to Alice using a randomly permuted vector of n ElGamal ciphertexts; among these n ciphertexts, the number of encryptions of g^{-1} should be equal to the output of Alice; all the remaining ciphertexts should be encryptions of g.

We construct S_2 as follows. S_2 simulates all internal coin flips of Bob as described in the protocol. S_2 simulates the message from Alice to Bob using n random ElGamal ciphertexts. The computational indistinguishability immediately follows from the semantic security of the ElGamal cryptosystem under the DDH assumption.

4.2 Privacy-Preserving Generalized Hamming Distance

In this protocol, we consider the case that $A = (a_1, \ldots, a_n)$ and $B = (b_1, \ldots, b_n)$ where each a_i and each b_i has a finite domain $\{1, ..., s\}$ rather than a binary domain. For these general discrete-valued a_i and b_i, we consider the Boolean difference function:

$$\mathsf{diff}(a_i, b_i) = \begin{cases} 0 \text{ if } \ a_i = b_i \\ 1 \text{ otherwise.} \end{cases}$$

We define the *generalized Hamming distance* as $\mathsf{gdist}(A, B) = \sum_{i=1}^{n} \mathsf{diff}(a_i, b_i)$. Our protocol for privately computing generalized Hamming distance, shown in Figure 2. Like the Hamming distance protocol, the generalized Hamming distance protocol also relies on the ElGamal cryptosystem. In this case, we take advantage of homomorphic properties obtained by encrypting in the exponent. That is, we encrypt a message m by using g^m as the cleartext in the ElGamal system rather than using m.

Theorem 2. *Under the DDH assumption, the protocol in Figure 2 for general discrete-valued inputs privately computes the generalized Hamming distance in the semi-honest model.*

Proof. We begin by showing correctness. For $1 \leq i \leq n$, for some random values k_i and ℓ_i, we have:

$$\begin{aligned} e''_i &= ({d_{i,1}}^{r_i}, {d_{i,2}}^{r_i}) \\ &= \left(\frac{{c'_{i,1}}^{r_i}}{{c_{i,1}}^{r_i}}, \frac{{c'_{i,2}}^{r_i}}{{c_{i,2}}^{r_i}} \right) \\ &= \left(\frac{g^{b_i} \cdot h^{k_i}}{g^{a_i} \cdot h^{\ell_i}}, \frac{g^{k_i}}{g^{\ell_i}} \right) \\ &= (g^{(b_i - a_i)} \cdot h^{(k_i - \ell_i)}, g^{(k_i - \ell_i)}). \end{aligned}$$

Privacy-Preserving Generalized Hamming Distance Protocol

Input: Vectors $A = (a_1, \ldots, a_n)$ and $B = (b_1, \ldots, b_n)$ held by Alice and Bob, respectively, such that $a_i, b_i \in \{1, \ldots, s\}$ for $1 \leq i \leq n$.

Output: Alice learns $\mathsf{gdist}(A, B)$.

1. For $1 \leq i \leq n$, Alice sends $E(g^{a_i}) = (c_{i,1}, c_{i,2})$ to Bob.
2. For $1 \leq i \leq n$, Bob computes $E(g^{b_i}) = (c'_{i,1}, c'_{i,2})$ and defines $e'_i = (c'_{i,1}/c_{i,1}, c'_{i,2}/c_{i,2}) = (d_{i,1}, d_{i,2})$. Then Bob chooses a random r_i and computes $e''_i = ((d_{i,1})^{r_i}, (d_{i,2})^{r_i})$. Bob sends a random permutation of all the e''_i to Alice.
3. Alice decrypts all received e''_i. Alice counts the total number of decryptions whose values are not equal to 1. This number is equal to $\sum_{i=1}^{n} \mathsf{diff}(a_i, b_i) = \mathsf{gdist}(A, B)$.

Fig. 2. Privacy-Preserving Generalized Hamming Distance Protocol

Thus, e''_i decrypts to $g^{(b_i - a_i)}$, which is equal to 1 if and only if $a_i = b_i$. It follows that the number Alice obtains in Step 3 of decryptions that are not equal to 1 is the desired distance.

To show privacy, we must show simulators S_1 for Alice and S_2 for Bob. S_1 simulates all internal coin flips of Alice as described in the protocol. S_1 simulates the message from Bob to Alice using a randomly permuted vector of n ElGamal ciphertexts. These n ciphertexts are chosen so that among these n ciphertexts, the number of encryptions of 1 is equal to the output of Alice; the remaining ciphertexts are encryptions of random cleartexts.

S_2 simulates all internal coin flips of Bob as described in the protocol. S_2 simulates the message from Alice to Bob using n random ElGamal ciphertexts.

The computational indistinguishability immediately follows from the semantic security of the ElGamal cryptosystem under the DDH assumption.

4.3 Experimental Results

We implemented these privacy-preserving Hamming distance and generalized Hamming distance protocols using the OpenSSL library in C. We carried out our experiments on a NetBSD machine with 2GHz CPU and 512M memory, using public keys of 1024 bits. The computation cost dominates the overall protocol, so we measured only the computation time for Alice and Bob. (That is, we did not measure the communication time.) We measured the computation cost of both the Hamming distance protocol and the generalized Hamming distance protocol on binary-valued vectors of varying lengths.

The results of our experiments are shown in Figure 3 for the Hamming distance protocol and Figure 4 for the generalized Hamming distance protocol. As expected, the experiments demonstrate that the computation time of Alice and Bob is linear in the vector size for both protocols. For the same length vectors,

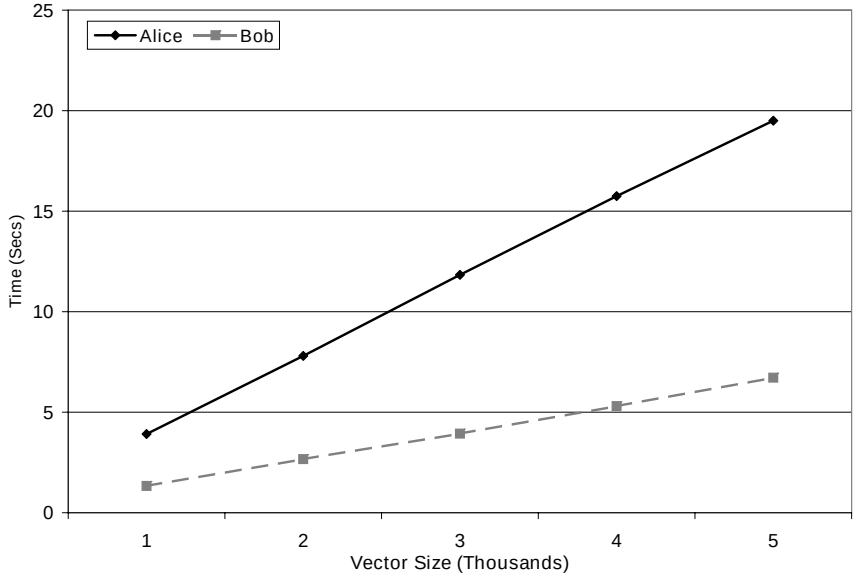

Fig. 3. Performance of Privacy-Preserving Hamming Distance Protocol

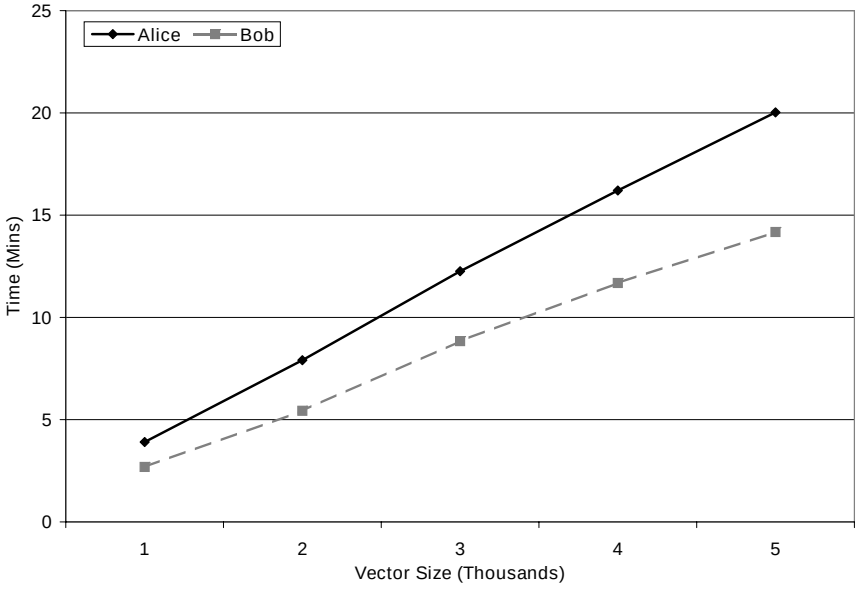

Fig. 4. Performance of Privacy-Preserving Generalized Hamming Distance Protocol

the generalized Hamming distance protocol takes Bob about twice as long to compute as the Hamming distance protocol.

5 Privacy-Preserving Model Selection

Many models have been proposed in the field of statistics, machine learning, and data mining, including linear models, neural networks, classification and regression trees, and kernel methods. One of the problems in data mining is how to select which kind of model is best for a particular task in a particular setting. Often, a human expert will make some initial judgment about which model or models seem likely candidates for the task at hand. With or without expert guidance, it is very common and useful to use measure the performance of a few learned models on test data to see which performs the best. Model selection is also useful for determining the parameters to use for a particular model, such as the depth of a decision tree.

In the setting of privacy-preserving data mining, it is important that the privacy that was maintained in learning data mining models is not lost in the model selection process. As mentioned earlier, in this paper we provide a first step towards a solution. We address a specific kind of vertical partitioning, in which one party holds all the data except the class labels and a second party holds all the class labels.

5.1 Problem Formulation

A database D is vertically partitioned between two parties, Alice and Bob. D contains n records in total. Alice's database D_A consists of m non-class attributes $(V_1, ..., V_m)$, where each V_i has a finite domain. Bob's database D_B includes only the class attribute C.

The parties want to collaborate to select an appropriate classification model to learn based on the combination of their databases—e.g., to decide whether to learn decision trees or naive Bayes classifiers from their data. However, because of privacy concerns, they do not wish to reveal their original data to each other.

Cross validation is a popular method for model selection. To carry out cross validation in a privacy-preserving manner, it is necessary to prevent the parties from learning anything they would not otherwise learn. In our setting, this means that Bob should not learn the predicted class labels for various models applied to Alice's data, and Alice should not learn the class labels that Bob has. In addition, candidate models should themselves be generated in a privacy-preserving way.

We present a privacy-preserving protocol for selecting a model on a database vertically partitioned so that Bob holds only the class labels. Specifically, we present a privacy-preserving protocol for selecting a model between decision trees and naive Bayes classifiers. Our protocol easily extends to any kind of classifier that can be learned in a privacy-preserving manner.

Privacy-preserving protocols for learning decision trees and for learning naive Bayes classifiers, respectively, have been proposed by Vaidya and Clifton [31,32].

Here, we use these two protocols as "black boxes" to achieve privacy-preserving model selection, using k-fold cross validation as an example. Our results extend straightforwardly to any type of model selection in which the choice of model depends only on the available data and the number of errors made by each model. This includes other types of cross validation such as the holdout method and leave-one-out cross validation.

In k-fold cross validation, both parties partition (their own parts of) the original database D into k pieces. The first $k-1$ pieces are *training sets* used to learn the model and the remaining piece is the *test set* used to validate or test the learned models. The parties learn both types of candidate model using a privacy-preserving protocol on each of the training sets (resulting in $k-1$ decision trees and $k-1$ naive Bayes classifiers). They then use the training set for estimating the classification error of each type of learned model. The type of model (i.e., decision tree or naive Bayes classifier) that has the lowest mean classification error over all $k-1$ learned models is considered the best, and is then learned (presumably again in a privacy-preserving manner) over the entire dataset. To compute the mean classification error for each type of model, we show how to compute the classification error for a single model in a privacy-preserving manner.

5.2 Our Protocol

After multiple classification models are learned from the database D for each of $k-1$ training sets, both parties need to compute the classification error for each model on the test set. Because Alice has all the non-class attributes, she can classify each record by herself using the classification model which has been learned. However, the classification error depends on the real class label which is held by Bob.

For privacy reasons, Alice cannot send the classification results to Bob, and Bob cannot send the actual class labels to Alice, as this would breach their privacy. Similarly, Alice cannot send her data to Bob so that he can apply the classifiers on her data and compare the results to the actual class labels. To get around this, we instead compute the classification errors using a privacy-preserving Hamming distance protocol (or its generalized version, if there are more than two possible class labels) such as the one presented in Section 3.

Alice's input to each instance of Hamming distance protocol is a vector $A = (a_1, ..., a_n)$, where each a_i is the class label predicted by Alice using the learned classification model, where n is the number of records in the test set. Bob's input is a vector $B = (b_1, ..., b_n)$, where each b_i is the real class label. From the resulting classification errors, Alice can determine the mean classification error for each type of model. We summarize the entire protocol in Figure 5.

Privacy-Preserving k-fold Cross Validation

Input: A database D vertically partitioned between Alice and Bob. Bob holds the class attribute and Alice holds all the other attributes.

Output: Alice and Bob learn the selected model for D.

1. Alice and Bob partition the database D into k pieces ($k-1$ training sets and one test set).
2. Alice and Bob use privacy-preserving protocols on the $k-1$ training sets to learn $k-1$ decision trees and $k-1$ naive Bayes classifiers.
3. For $1 \leq i \leq k-1$, Alice and Bob carry out the following steps:
   (a) Alice classifies her records in the test set using the ith learned decision tree and naive Bayes classifier.
   (b) Alice and Bob use the privacy-preserving Hamming distance protocol (or generalized Hamming distance protocol, as appropriate) for Alice to compute the classification error from the ith learned decision tree and from the ith learned naive Bayes classifier on the test set. (That is, comparing her results computed in Step 3a to Bob's actual class values in the test set.)
4. Combining the $k-1$ results, Alice computes the mean classification error for the decision tree and for the naive Bayes classifier, and announces these results to Bob. Alice and Bob then select the type of model which has the lower mean classification error.

Fig. 5. Privacy-Preserving Model Selection Protocol

As we have described it in Figure 5, the protocol leaks partial information in step 3b—namely, the number of misclassified records for each model on the test set. However, if desired, one could remove this leak by stopping all of the generalized Hamming distance protocols before Bob sends the the final results to Alice, and using a Yao secure two-party computation on Bob's encryptions and Alice's decryption key for the parties to learn only the mean classification error. However, this step would add substantial additional cost unless the total number of records in the test set and total number of training sets are relatively small.

6 Discussion

In this paper, we introduced privacy-preserving model selection. This is important for "extending the boundary" of privacy-preserving protocols to include steps beyond the computation of particular data mining models. By extending the boundary of what can be accomplished with efficient privacy-preserving computation, we bring the adoption of privacy-preserving data mining closer to practice.

Our privacy-preserving solution enables model selection via cross validation on a database vertically partitioned between two parties. Our solution is based on a privacy-preserving primitive for computing the Hamming distance or generalized Hamming distance of two vectors, which may be of independent interest.

There are a number of interesting directions yet to be studied. Most importantly, our setting considers a very extreme case of partitioning. We argue this is realistic in some cases, but clearly it is not applicable in all cases. Additionally, in order to provide the greatest privacy protection during the model selection process not only the raw data, but also the candidate models that are not selected should be kept private, as revealing multiple models provides more information about the data than revealing just the selected type of model does.

An attractive option, that could both allow more general vertical partitioning and protect privacy further by not releasing the individual candidate models, is to have the models themselves be computed in such a way that they are not known to the individual parties, but can be used by them through yet another protocol. To our knowledge, only a small number of privacy-preserving protocols do this—namely, Vaidya and Clifton's naive Bayes classifier protocol [31] and Laur, Lipmaa and Mielikäinen's support vector machines [24]. By using such protocols and modifying our Hamming distance protocol to have Alice's input shared by Alice and Bob instead of known to Alice, it should be possible to obtain a solution in which the parties perform model selection without revealing the candidate models considered. This kind of solution would also be appealing because it can maintain the privacy of the classifier results even in the case that both parties know the real class labels.

Our proposed protocol defends against semi-honest adversaries. It is open to extend them to efficient protocols that provide security against malicious adversaries. It might also be interesting to consider other distance metrics such as the L1-distance that allow for finer granularity by considering some wrong answers more acceptable than others. Finally, as the privacy definitions in secure multiparty computation are very strict, relaxed yet meaningful privacy definitions that enable more practical protocols deserve further exploration.

Acknowledgments

We thank the attendees of the PinKDD'07 workshop for helpful and interesting discussions.

References

1. D. Agrawal and C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proc. of the 20th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 247–255, 2001.
2. R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proc. of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 86–97, 2003.

3. R. Agrawal and R. Srikant. Privacy preserving data mining. In *Proc. of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 439–450, May 2000.
4. M. Atallah and W. Du. Secure multi-party computational geometry. In *Proc. of the Seventh International Workshop on Algorithms and Data Structures*, pages 165–179. Springer-Verlag, 2001.
5. D. Boneh. The decision Diffie-Hellman problem. In *ANTS-III*, volume 1423 of *LNCS*, pages 48–63, 1998.
6. D. Boneh, E. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Prof. of the Second Theory of Cryptography Conference*, volume 3378 of *LNCS*. Springer-Verlag, 2005.
7. R. Canetti, Y. Ishai, R. Kumar, M. Reiter, R. Rubinfeld, and R. Wright. Selective private function evaluation with applications to private statistics. In *Proc. of the 20th Annual ACM Symposium on Principles of Distributed Computing*, pages 293–304, 2001.
8. W. Du and Z. Zhan. Using randomized response techniques for privacy-preserving data mining. In *Proc. of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 505–510, 2003.
9. T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4), 1985.
10. A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proc. of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 211–222, 2003.
11. A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proc. of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–228, 2002.
12. J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. Strauss, and R. Wright. Secure multiparty computation of approximations. *ACM Transactions on Algorithms*, 2(3):435–472, 2005.
13. M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 1–19. Springer-Verlag, 2004.
14. B. Goethals, S. Laur, H. Lipmaa, and T. Mielikäinen. On private scalar product computation for privacy-preserving data mining. In *Proc. of the Seventh Annual International Conference in Information Security and Cryptology*, volume 3506 of *LNCS*. Springer-Verlag, 2004.
15. O. Goldreich. *Foundations of Cryptography, Volume II: Basic Applications*. Cambridge University Press, 2004.
16. O. Goldreich, S. Micali, and A. Wigderson. How to play ANY mental game. In *Proc. of the 19th Annual ACM Conference on Theory of Computing*, pages 218–229, 1987.
17. Z. Huang, W. Du, and B. Chen. Deriving private information from randomized data. In *Proceedings of the ACM SIGMOD Conference*, 2005.
18. P. Indyk and D. Woodruff. Polylogarithmic private approximations and efficient matching. In *Prof. of the Third Theory of Cryptography Conference*, LNCS. Springer-Verlag, 2006.
19. G. Jagannathan and R. N. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proc. of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 593–599, 2005.

20. G. Jagannathan and R. N. Wright. Privacy-preserving data imputation. In *Proc. of the ICDM Int. Workshop on Privacy Aspects of Data Mining*, pages 535–540, 2006.
21. M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *Proc. of the ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'02)*, pages 24–31, June 2002.
22. M. Kantarcioglu and J. Vaidya. Privacy preserving naive Bayes classifier for horizontally partitioned data. In *IEEE Workshop on Privacy Preserving Data Mining*, 2003.
23. H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *The Third IEEE International Conference on Data Mining*, 2003.
24. S. Laur, H. Lipmaa, and T. Mielikäinen. Cryptographically private support vector machines. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 618–624, 2006.
25. Y. Lindell and B. Pinkas. Privacy preserving data mining. *J. Cryptology*, 15(3):177–206, 2002.
26. K. Liu, H. Kargupta, and J. Ryan. Multiplicative noise, random projection, and privacy preserving data mining from distributed multi-party data. Technical Report TR-CS-03-24, Computer Science and Electrical Engineering Department, University of Maryland, Baltimore County, 2003.
27. D. Meng, K. Sivakumar, and H. Kargupta. Privacy-sensitive Bayesian network parameter learning. In *Proc. of the Fourth IEEE International Conference on Data Mining*, Brighton, UK, 2004.
28. S. Rizvi and J. Haritsa. Maintaining data privacy in association rule mining. In *Proc. of the 28th VLDB Conference*, 2002.
29. J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proc. of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 639–644, 2002.
30. J. Vaidya and C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proc. of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 206–215, 2003.
31. J. Vaidya and C. Clifton. Privacy preserving naive Bayes classifier on vertically partitioned data. In *2004 SIAM International Conference on Data Mining*, 2004.
32. J. Vaidya and C. Clifton. Privacy-preserving decision trees over vertically partitioned data. In *The 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, 2005.
33. Z. Yang, H. Subramaniam, and R. N. Wright. Experimental analysis of a privacy-preserving scalar product protocol. *International Journal of Computer Systems Science and Engineering*, 21(1):47–52, 2006.
34. Z. Yang and R. Wright. Privacy-preserving computation of Bayesian networks on vertically partitioned data. *IEEE Transactions on Data Knowledge Engineering*, 18(9), 2006.
35. A. Yao. How to generate and exchange secrets. In *Proc. of the 27th IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.