PerturBoost: Practical Confidential Classifier Learning in the Cloud

Keke Chen, Shumin Guo
Data Intensive Analysis and Computing (DIAC) Lab, Department of Computer Science and Engineering
Wright State University, Dayton, OH 45435, Email: {keke.chen,guo.18}@wright.edu

***Abstract*—Mining large data requires intensive computing resources and data mining expertise, which might not be available for many users. With the development of cloud computing and services computing, data mining tasks can now be moved to the cloud or outsourced to third parties to save costs. In this new paradigm, data and model confidentiality becomes the major concern to the data owner. Meanwhile, users are also concerned about the potential tradeoff among costs, model quality, and confidentiality. In this paper, we propose the PerturBoost framework to address the problems in confidential cloud or outsourced learning. PerturBoost combined with the random space perturbation (RASP) method that was also developed by us can effectively protect data confidentiality, model confidentiality, and model quality with low client-side costs. Based on the boosting framework, we develop a number of base learner algorithms that can learn linear classifiers from the RASP-perturbed data. This approach has been evaluated with public datasets. The result shows that the RASP-based PerturBoost can provide model accuracy very close to the classifiers trained with the original data and the AdaBoost method, with high confidentiality guarantee and acceptable costs.**

I. Introduction

Most data mining tasks require a good understanding of the mining techniques, time-consuming parameter tuning, algorithm tweaking, and sometimes algorithm innovation. They are often resource-intensive and may need the expertise of applying data-mining techniques in a large-scale parallel processing cluster. As a result, many data owners, who have no sufficient computing resources or data-mining expertise, cannot mine their data by themselves.

The development of cloud computing and services computing enables at least two solutions. First, if the data owner has the data-mining expertise but not the computing resources, he/she can rent public cloud resources to process the data with low cost (e.g., a 100-node cluster built with small virtual machine instances in Amazon Cloud costs only $8 per hour). Second, if the data owner does not have the expertise, he/she can outsource their data-mining tasks to data-mining service providers. Many companies such as Kaggle (kaggle.com) have started providing such data mining services[1].

In spite of the tremendous benefits in cost saving, the unprotected outsourcing approach has three major drawbacks. (1) The exported data may contain private information, which is the reason that Netflix suspended the Netflix prize II competition [11]. (2) The data ownership is not protected. Once published, the dataset can be accessed and used by any one. (3)The ownership of the resultant models is not protected. At least the model developer knows the model and understands how to use it. The developer can possibly use the model by himself/herself or share it with others without the data owner's permission.

In addition to confidentiality, in cloud computing, the client-side cost is an important factor in data owner's decision making. If the client-side cost is much higher than in-house mining, the data owner may not consider using cloud resources. Specifically, such client-side costs include in-house pre- and post- processing, and the communication between the client and the cloud. A practical solution has to consider minimizing the client-side costs.

Note that these features are distinct from the previous studies on *shared-data/model privacy preserving data mining*, which focus on sharing data and models without leaking the private information in the data. Thus, methods like data anonymization [6] are applicable. It is also different from multi-party privacy preserving data mining [9], where each party shares some data but does not want other parties to find out the private information in the shared data. However, it requires local data processing, which contradicts the purpose of cloud computing.

Our Approach. Preserving data/model confidentiality often impairs data utility, which contradicts the goal of learning high quality models. We propose the PerturBoost framework to address these contradicting goals. The current work is focused on learning classifiers. This approach has a number of unique features and contributions.

1) It utilizes the random space perturbation (RASP) method proposed by our previous work [3] to protect data confidentiality, which has been proven secure in outsourced database services [13]. It provides much higher level of confidentiality compared to existing perturbation methods such as geometric perturbation [4] and random projection perturbation [10].
2) We design several methods to securely learn classifiers from RASP perturbed data. The RASP approach was originally designed only for confidential database query processing. By extending the secure query processing method, we are able to securely learn linear classifiers from the perturbed data, and meanwhile preserve the confidentiality of model.
3) Simple linear classifiers do not provide high prediction

[1]As both infrastructure services and mining services are treated as cloud services in a broader definition, we will use cloud-based mining in this paper.

accuracy. We extend the learning methods with the boosting framework to promote model accuracy. We show that the boosting framework works nicely and the result is very close to the classifiers learned with the original data.

4) To minimize the client-side costs, we develop two base-classifer algorithms Greedy Decision Stump and Greedy Linear Classifier that allow the cloud side to derive good base classifiers asynchronously with only the initial batch of "seed classifiers" provided by the client. The model quality is also well maintained with these cost-effective methods.

The proposed framework can be easily extended to other perturbation methods, such as random projection perturbation (RPP) [10] and geometric perturbation (GDP) [4]. Due to the space limit, we do not include the details on the extended study.

The remaining sections are organized as follows. We will briefly review the related work in Section II. Section III gives the notations and the background knowledge. Section IV presents the PerturBoost framework and the RASP-related learning methods in detail. Section V shows the experimental study.

II. Related Work

Shared data or model privacy-preserving data mining (PPDM) is probably the closest work to confidential mining in the cloud, which includes three groups of techniques. (1) Additive perturbation techniques that hide the real values by adding noises [2]. Because the resultant models are not protected, they are not appropriate for outsourced mining. (2) Cryptographic protocols enabling multiparty collaborative mining without leaking either party's private information [9]. These protocols expect the participants process the data in house - a principle contradicting outsourced computation. (3) Data anonymization [6] that disguise personal identities or virtual identifiers in the shared data. However, it does not protect sensitive attributes and the resultant models.

Secure database outsourcing has a similar setting to cloud mining. In secure outsourced database, the major database components, basically indexing and query processing, are moved to the cloud. Typical techniques include order preserving encryption (OPE) [1], crypto-index [8], and RASP [3].

Fully homomorphic encryption [7] envisions an ideal scenario for confidential cloud computing. Theoretically, once the basic homomorphic addition and multiplication are implemented, any functions can be derived with the basic operations. However, the current solutions are still too expensive to be used in practice [12].

III. Preliminary

First, we will give the notations and basic concepts used by this paper. Then, we will also briefly introduce the RASP perturbation method to make the paper self-contained.

A. Notations

Our work will be focused on classifier learning on numeric datasets. Classifier learning is to learn a model $y = f(x)$ from a set of training examples $R = \{(x_i, y_i), i = 1 \dots N\}$, where N is the number of examples, $x_i \in \mathbb{R}^k$ is a k-dimensional feature vectors describing an example, and y_i is the label for the example - if we use '+1' and '-1' to indicate two classes, $y_i \in \{-1, +1\}$. The learning result is a function $y = f(x)$, i.e., given any known feature vector x, we can predict the label y for the example x. The quality of the model is defined as the accuracy of prediction on the testing set T.

We will use the boosting framework [5] in our approach. A boosted model is a weighted sum of n base classifiers, $H(x) = \sum_{i=1}^{n} \alpha_i h_i(x)$, where (1) the base models $h_i(x)$ can be any *weak learner*, e.g., a learner with its accuracy significantly higher than 50% for two-class prediction as the accuracy of a random guess to the two-class problem would be around 50%; and (2) α_i, $\alpha_i \in \mathbb{R}$, are the weights of the base models, which are learned using algorithms such as AdaBoost [5].

B. RASP perturbation

RASP works on vector data. For each k-dimensional original vector x_i, the RASP perturbation can be described in the following formula.

$$\begin{aligned} z_i &= RASP(x_i; A, K_{OPE}) \\ &= A(E_{OPE}(K_{OPE}, x_i)^T, 1, v_i)^T, \end{aligned} \quad (1)$$

where (1) z_i is the perturbation result, a $k+2$ dimensional vector; (2) E_{OPE} is an order preserving encryption[2] (OPE) [1] with encryption key K_{OPE}. The OPE scheme is used to transform the distribution of j-th dimension X_j to the standard normal distribution with dimensional order preserved. (3) v_i is drawn from the standard normal distribution, with $v_i > v_0$, where v_0 is a constant so that the probability of having $v_i < v_0$ is negligible. (4) A is a $(k+2) \times (k+2)$ randomly generated invertible matrix. A is the secret key matrix shared by all vectors, but v_i is randomly generated for each individual vector. As a result, the same x_i can be mapped to different z_i in the perturbed space due to the randomly chosen v_i, which provides extra protection. We have proved that RASP is neither distance preserving nor order preserving [13].

Secure Half-space Query. The RASP perturbation approach enables a secure query transformation and processing method for half-space queries [3]. A simple half-space query like $X_i < a$, where X_i represents the dimension i and a is a scalar, can be transformed to an encrypted half-space query in the perturbed space: $z_i^T Q z_i < 0$, where z_i is the perturbed vector, and Q is a $(k+2) \times (k+2)$ *query matrix*. Specifically,

$$Q = (A^{-1})^T v u^T A^{-1}, \quad (2)$$

where $u = (w^T, 1, -E_{OPE}(a))^T$, w is the dimension indication vector: all entries are zero except for the dimension i set to 1, and $v = (0, \dots, -1, E_{OPE}(v_0))$ is a vector with all

[2]E.g., if $x < y$, then $E_{OPE}(x) < E_{OPE}(y)$.

entries zero except for the last two. This quadratic query form $z_i^T Q z_i < 0$ is derived from the equivalent query condition $(X_i - a)(z_{k+2} - v_0) < 0$, where z_{k+2} is the appended noise dimension with guaranteed $z_{k+2} - v_0 > 0$. The basic idea is to transform each of the original conditions, say $X_i < a$ to a general vector form $u^T A^{-1} z < 0$, and $z_{k+1} - v_0 > 0$ to $(z^T A^{-1})^T v > 0$. With Eq. 1, it is easy to verify the above transformation is correct. As long as the matrix A keeps confidential, there is no effective method to recover the condition $X_i < a$ from the exposed matrix Q.

C. Security Assumptions

In the practical cloud environment, it is appropriate to assume that the service providers are honest-but-curious parties, who will honestly provide the services but may want to peek at or resell the data owner's private data. The data owner exports the data and receives the mined models. Curious service providers can see the outsourced data, each execution step of the mining algorithm, and the generated model.

There are two levels of adversarial prior knowledge. (1) If the user only uses the cloud infrastructure for mining, we can safely assume the adversaries know only the perturbed data, corresponding to the ciphertext-only attack in cryptoanalysis. (2) In the case that mining services are used, in addition to the perturbed data, we also assume the adversaries know the feature distributions, as such information might be provided for model analysis or exposed via other channels to the service provider. We exclude the case of insider attacks, e.g., an insider on the user side colludes with the adversary and provides perturbation details or original unperturbed data records, which will be extremely difficult to handle.

We define data confidentiality as the resilience of the protected data to any data reconstruction or estimation methods, which can be effectively evaluated with the mean-squared-error (MSE) approach [4]. Assume $\{\hat{v}_i, i = 1..N\}$ is a series of estimated values for the original data $\{v_i, i = 1..N\}$. Let g_t be an estimation method in the set of all possible methods $\mathcal{G}$. Then, the level of preserved confidentiality can be evaluated by the measure $\gamma = \min_{g_t \in \mathcal{G}} 1/N \sum_{i=1}^{N} (\hat{v}_i - v_i)^2$.

Model confidentiality can be evaluated in a similar way. We assume the adversary knows what type of model is being developed, because such information cannot be effectively hidden. Therefore, model confidentiality means the confidentiality of the model parameters. For example, for linear classifiers, $f(x) = w^T x + b$, the confidentiality of the parameters w and b are important to preserve. Similarly, we can define model confidentiality as the estimation accuracy of the model parameters.

IV. The PerturBoost Framework

Firstly, we will briefly describe the procedure of learning with the cloud or the mining service provider. Then, we will discuss the key algorithms for the RASP-based PerturBoost approach.

Preparing Training Data. The user uses the RASP perturbation to prepare the training data for outsourcing. To protect the confidentiality of training data, we assume it is sufficient to protect the confidentiality of the feature vectors x_i of each training record $\{x_i, y_i\}$, while leaving y_i unchanged. This exposure will leave very limited information to the attackers. Furthermore, if the user can proportionally sample the data to be outsourced to generate uniform label distribution, no label information will be leaked. For example, in two-class problems, the user can prepare about the same number of examples for each class. Now the problem becomes learning from the data $\{(RASP(x_i), y_i)\}$.

Securely Learning Models. To make sure the models learned from the perturbed data useful, we introduce the definition of ϵ-effective learning. Let H be the classifier learned from the original data $\{(x_i, y_i)\}$ and H_P be the one learned from $\{(P(x_i), y_i)\}$, where $P()$ is a specific perturbation method.

Definition 1: Let $Error()$ represent the classification evaluation function. For any set of testing data, if $|Error(H) - Error(H_P)| < \epsilon$, where ϵ is a user-defined small positive number, we say that learning from the perturbed data is ϵ-*effective*.

In practice, because of the downgraded data quality (e.g., noise addition) or the specific way transforming the data, the available learning methods are quite limited and learning from perturbed data often results in sub-optimal models. To find ϵ-effective classifiers for small ϵ, we try to incorporate the boosting idea in the PerturBoost framework. The PerturBoost framework extends the existing boosting algorithm such as AdaBoost [5], and generates models in the following form.

$$H_P = \sum_{i=1}^{n} \alpha_i h_P^{(i)}, \tag{3}$$

where $h_P^{(i)}, i = 1..n$, are the models *learned from the perturbed data with special base learners*. Thus, the key challenge is developing the base-learner algorithms that can learn from RASP-perturbed data. We will depend on experimental study to evaluate the effectiveness of the PerturBoost framework with the special base learners.

Applying Learned Models. There are different ways to apply the mined models. The mined model, say $M = H_P(P(x))$, should be returned in some protected form so that adversaries cannot take advantage of it. Let D_{new} be the new dataset. When the user applies the model, two methods are available: (1)either perturbing the data: $D'_{new} = P(D_{new})$ and then applying $H_P(D'_{new})$, or (2) recovering the model in the original space: $Transform(M) \rightarrow M' = H'(D)$ and then applying $H'(D_{new})$. Obviously, the model recovering approach is more cost-effective. However, in the situation that the model cannot be easily recovered, the first approach will have to be applied.

A. Securely Learning Classifiers from RASP-Perturbed Data

With the PerturBoost framework, the key is the base-learner algorithms that can learn from the perturbed data. In this section, we focus on the learning algorithms for RASP-perturbed data. Specifically, we will develop linear classifier

learning algorithms, based on the RASP secure query processing method.

Query-based Linear Classifiers. As we have shown in Section III, half-space queries can be transformed to the RASP perturbed space and be processed securely. Specifically, a half-space condition like $X_i < a$ is transformed to the condition $z^T Q z < 0$, where transformation is done by the user, who provides the query matrix Q to the server. Because the labels are unchanged, it is possible to count the number of '+1' and '-1' examples on the half-space, respectively, with the following query:

select count(y="+1"), count (y="-1")
from P=$\{z_i$=RASP$(x_i), y_i\}$
where $z^T Q z < 0$.

We can similarly derive the number of '+1' and '-1' examples on the other half space with the condition $z^T Q z \geq 0$. Then, the half-space can be used to define a classifier, such as

$$f(z) = z^T Q z \begin{cases} < 0, return -1 \\ \geq 0, return +1 \end{cases} \quad (4)$$

which has prediction error lower than 50%[3]. This is actually a decision stump (DS) [5] in the perturbed space.

Random Decision Stump (RandomDS). According to the Adaboost algorithm [5], in each round, the algorithm needs to find the classifier h_t in the family of weak classifiers $\mathcal{H}$ that maximizes the absolute value of the difference of the corresponding weighted error rate ε_t and 0.5:

$$h_t = \arg\max_{h_t \in \mathcal{H}} |\varepsilon_t - 0.5|, \text{where } \varepsilon_t = \sum_{i=1}^{N} w_i h_t(z_i).$$

Specifically, if decision stump is used as the weak learner, the algorithm will scan through all possible splitting values for all dimensions to find the best decision stump. This becomes prohibitively expensive for the RASP-query based decision stumps - the user cannot afford encoding all possible decision stumps and sending them to the service provider.

Instead, we propose to use a random sampling method: the *RandomDS* method, to reduce the cost of finding a *good* decision stump. Specifically, in each round, the service provider asks the client to provide a random set of decision stumps that approximates the set $\mathcal{H}$, denoted as $\tilde{\mathcal{H}}$. The sub-optimal decision stump is found as the weak learner.

How to appropriately sample the decision stumps? Because each dimension has been transformed to the standard normal distribution in the OPE step, we can sample $-E_{OPE}(a)$ from the central range of the distribution, say [-2, 2], the contains the majority of the population.

Random Linear Classifier (RandomLC). The decision stump method can be extended to the more general case - the linear classifiers. Remember that a one-dimensional half-space, e.g., $X_i < a$, is first transformed to the general half-space representation $u^T z < 0$, where $u = (w^T, 1, -E_{OPE}(a))^T$ and w is the dimension indicating vector. In this transformation, we can also arbitrarily chose w so that a general half-space $w^T E_{OPE}(x) - E_{OPE}(a) < 0$ in the OPE transformed space is generated. Using the same method for deriving the decision stump classifier, we can then get a general linear classifier as the base learner.

As there are an unlimited number of linear classifiers, we have to use the sampling approach again to provide a small subset of linear classifiers in each round. One of the critical problem is to effectively sample the parameter space of w and $E_{OPE}(a)$, so that the limited sample chances will not be wasted on low-accuracy models.

The basic idea is to find the hyperplanes that "shatter" the center of the dataset. Because the OPE transformed dimensions have the standard normal distribution, we design the following sampling method.

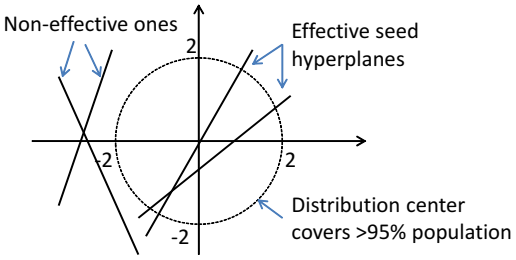

Fig. 1. Effective seed hyperplanes shatter around the distribution center.

As Figure 1 shows the two dimensional case, the majority of the records in two dimensional standard normal distribution will be enclosed in a circle of radius $r = 2$ and centered on (0, 0). To shatter around the center (0, 0), we need to have the distance between the center and the hyperplane less than the radius, e.g.,

$$\frac{|E_{OPE}(a)|}{|w|} < r. \quad (5)$$

We can generate a random unit vector w so that $|w| = 1$, and then we only need to find a random sample α from the standard normal distribution as $E_{OPE}(a)$ that satisfies $|\alpha| < r$.

B. Asynchronous Learning Algorithms

Note that with the RandomDS and RandomLC algorithms the client side needs to stay alive to participate in each round of boosting iterations, which is expensive and inconvenient to users. In the following we discuss two asynchronous learning algorithms that only request the user to submit a set of initial settings together with the perturbed data in the beginning - no need to participate in the iterations.

Greedy Decision Stump (GreedyDS). The basic idea is to utilize the property of the RASP query transformation method to derive an arbitrary number of decision stumps based on a small set of seed decision stumps.

Let a and b be two arbitrary different values on the dimension X_i, then any point on the dimension can be represented as $x = a + \lambda(b - a)$, where $\lambda \in \mathbb{R}$. We show that

Proposition 1: Assume the query matrices Q_a and Q_b encode the conditions $X_i < a$ and $X_i < b$, respectively.

[3]reverse the prediction, if the error rate $> 50\%$.

Then, $Q_x = Q_a + \lambda(Q_b - Q_a)$, where λ is some real number, represents a valid threshold condition on X_i.

Proof: According to Eq. 2, Let $Q_a = (A^{-1})^T u_a v^T A^{-1}$ and $Q_b = (A^{-1})^T u_b v^T A^{-1}$. We have $Q_a + \lambda(Q_b - Q_a) = (A^{-1})^T(u_a + \lambda(u_b - u_a))v^T A^{-1}$. According to the definition of u vector, we have $u_a + \lambda(u_b - u_a) = (w^T, 1, -(E_{OPE}(a) + \lambda(E_{OPE}(b) - E_{OPE}(a))))^T$. It is easy to verify that $E_{OPE}(a) + \lambda(E_{OPE}(b) - E_{OPE}(a))$ is a valid value on the OPE transformed dimension. ■

Therefore, the client only needs to prepare two seed decision stumps per dimension. The server can derive an arbitrary number of decision stumps based on the seeds. A greedy method can be applied to find the best one among the candidates. However, not all of these decision stumps are effective, which waste the server computing time. Again, we hope they will shatter around the center of the population. We can achieve this by setting the seeds around the bounds [-2, 2]. Specifically, we can properly set the lower bound as the condition $E_{OPE}(a) < z_i$ and the upper bound $z_i < E_{OPE}(b)$.

Greedy Linear Classifier (GreedyLC). The greedy search algorithm can also be applied to general linear classifiers. Similarly, we have the following Proposition.

Proposition 2: Assume the query matrices Q_a and Q_b encode the general half-space conditions $u_a z < 0$ and $u_b z < 0$, respectively. Then, $Q_x = Q_a + \lambda(Q_b - Q_a)$, where λ is some real number, represents a valid general half-space condition.

The proof is similar to Proposition 1. Thus, we skip the details.

With the greedy linear classifier algorithms, the client will only generate the initial batch of m seed classifiers and send them to the server. In each iteration, the server will use an algorithm, according to Proposition 2, to derive new linear classifiers in a greedy manner.

C. Model Confidentiality

The resultant model consists of two components: the unprotected weights α_t, $t = 1..n$, and the protected base classifiers $h_t(F(x))$. Let's check whether the confidentiality of $h_t(F(x))$ is sufficiently protected.

Note that each $h_t(F(x))$ in the proposed methods corresponds to a protected half-space query. As proved by Xu et al. [13], under the security model described in Section III and without any additional information leaking, the query confidentiality is preserved.

To ensure whether the proposed algorithms result in confidential models, we should carefully check *weather there is no additional information leaking when the base classifiers are generated*. Note that the three algorithms: RandomDS, RandomLC, and GreedyLC use randomly generated base classifier set $\tilde{\mathcal{H}}$. The whole set $\mathcal{H}$ for either decision stumps or general linear classifiers has a prohibitively large number of members. A brute-force attack will need to enumerate all the possible members, which is computationally intractable. Therefore, the confidentially of these three methods are well preserved.

However, GreedyDS has some weaknesses on security, especially, when the adversary knows the original data distribution and the seed selection method. If the corresponding thresholds $E_{OPE}(a)$ and $E_{OPE}(b)$ are known, e.g., around -2 and 2, at each step of greedy algorithm, the adversary can infer the corresponding threshold of the generated decision stump in the OPE transformed space, with the server selected value λ. If the original data distribution is known, it is not difficult to map the OPE space back to the original data space, and thus the resultant models can be well estimated. An effective remedy is to relax the selection of the seeds decision stumps to purely random selection, which, however, may reduce the effectiveness of the consequently generated decision stumps.

V. Experiments

The previous sections have addressed the three major aspects: data confidentiality, model confidentiality, and the client-side costs. Specifically, we use the RASP perturbation to protect data confidentiality; model confidentiality is protected by the secure RASP query transformation method and the random selection of seed classifiers; we also develop two algorithms Greedy Decision Stump and Greedy Linear Classifier to reduce the client-side costs. The experiments will focus on the two aspects: the client-side costs and model accuracy. Due to the page limit, we skip the experiments on seed classifiers and combining PerturBoost and other perturbation methods.

A. Experiment Setup

Datasets. For easier validation and reproducibility of our results, we use a set of public datasets for evaluation, which have only two labeled classes, from UCI machine learning repository. These datasets have been widely applied in various classification modeling and evaluation.

In pre-processing, the missing values in some datasets (e.g., the Breast-Cancer and Ionosphere datasets) are replaced with random samples from the domain of the corresponding dimension. They are then normalized with the transformation $(v - \mu_j)/\sigma_j$, where μ_j is the mean and σ_j^2 is the variance of the dimension j, to minimize the differences on dimensional distributions. Each of the datasets is also perturbed with RASP. Then, the datasets are randomly shuffled and split for single-split evaluation and for five-fold cross validation.

Implementation. We implement the perturbation methods based on the algorithms in the paper [3]. The PerturBoost framework is implemented based on the Adaboost algorithm [5]. The four RASP-based base learners are implemented as plugins to the framework. All these implementations use C++ and are thoroughly tested on a Ubuntu linux server. We also used the Weka (www.cs.waikato.ac.nz/ml/weka/) implementation of Adaboost to generate the baseline accuracy using the original datasets.

B. Experimental Result

Cost of Preparing Base Learners. Users of cloud computing and services computing are also concerned with the client-side costs. We conduct a simple evaluation to show the expected costs for the RASP based methods.

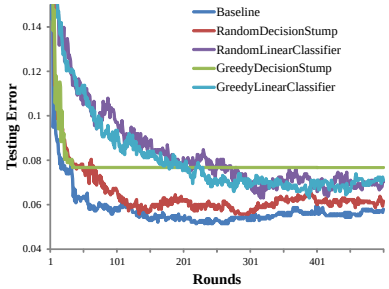

Fig. 2. Progressive testing error on Spambase, 500 rounds, 100 random DS/LS per batch.

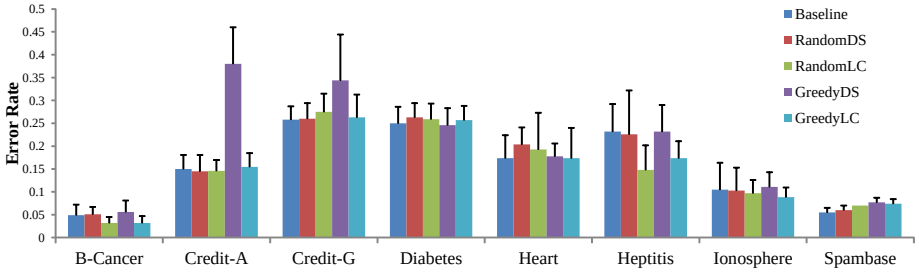

Fig. 3. 5-fold cross-validation results on all datasets, 500 rounds, 100 random queries per batch.

	Setting	Preparing Time	Transfer Bytes
Single Query	57dimensions	3.736 ms	26KB
RandomDS /LC	per round	∼22 sec	148MB
GreedyDS	one time	∼0.45 sec	3MB
GreedyLC	one time	∼22 sec	148MB

TABLE I
AVERAGE COST FOR DIFFERENT METHODS ON SPAMBASE.

	Baseline	RandomDS	GreedyDS	RandomLC	GreedyLC
Error rate(%)	5.5±1.2	6.0±1.1	7.7±1.4	7.1±1.0	7.4 ±1.0

TABLE II
CROSS-VALIDATION RESULTS ON SPAMBASE.

The client-side costs for RASP based methods basically include the costs generating the transformed queries and transferring them to the service provider. We use the amount of data to be transferred to represent the communication cost. This experiment is done on the Spambase dataset, which has 57 dimensions. The first line of Table I lists the per-query average cost based on preparing 1000 randomly generated queries, and the number of bytes to be transferred per query to the server. Overall, if d is the number of dimensions, the preparing time is approximately proportional to d^3 and the transfer bytes is proportional to d^2.

Based on the per-query costs we can derive the client-side costs for different RASP-based methods. Here we use 100 seed queries per dimension per round for RandomDS and RandomLC, and 100 seed queries per dimension for GreedyLC. The GreedyDS needs only two seed queries per dimension. RandomDS/LC have huge transfer costs, probably not good for high-dimensional data like this. GreedyLC has only moderate one-time cost. In contrast, GreedyDS has impressively very low one-time cost.

Effectiveness of Boosting. First, we look at the detailed boosting result on single-split training. The records in Spambase are shuffled and split into the training dataset (70%) and the testing dataet (30%). Figure 2 shows the progressive testing error in 500 rounds. The RandomDS/LS algorithms use 100 random queries per round. The GreedyLC algorithm uses 100 random queries for the one-time setup.

The result shows that RandomDS has the best performance. It converges fast and the accuracy is very close to the baseline. RandomLC and GreedyLC have similar performance; thus, we have no reason to use the more expensive RandomLC. GreedyDS converges fast, but it stays at the highest error rate, which probably traps in the local minima due to the less choices of the available base classifiers. However, its low client-side cost still makes it a competitive choice. Overall, the results on Spambase are very close, located in the range of error rate (0.05, 0.08). The five-fold cross-validation (Table II) also supports this observation. Although the average error rates are different, the standard deviations overlap each other, which implies that no method is statistically significantly better or worse than another in accuracy.

The results on other datasets show slightly different patterns. Figure 3 shows the five-fold cross-validation results for all the experimental datasets. The four methods except for GreedyDS perform similarly and consistently. GreedyDS has much higher error rates on Gredit-A and Credit-G. It is consistent with the conclusion we have drawn from the Spambase data.

VI. CONCLUSION

This paper presents the PerturBoost approach that aims to provide practical confidential classifier learning in the cloud. Such a practical confidential learning method should address the problems in four aspects: data confidentiality, model confidentiality, model quality, and low client-side costs. Specifically, we focus on the RASP-perturbation based methods that provide good data confidentiality and model confidentiality. We develop methods to show that weak linear classifiers can be learned from the RASP perturbed data. These weak linear classifiers are plugged into the PerturBoost framework to gain high-quality classifiers without breaching the model confidentiality. We also design learning methods to minimize the client-side costs and enable asynchronous learning without intensive client-cloud interactions. The experimental results show that PerturBoost can robustly restore the model quality for RASP-perturbed data.

REFERENCES

[1] AGRAWAL, R., KIERNAN, J., SRIKANT, R., AND XU, Y. Order preserving encryption for numeric data. In *Proceedings of ACM SIGMOD Conference* (2004).

[2] AGRAWAL, R., AND SRIKANT, R. Privacy-preserving data mining. In *Proceedings of ACM SIGMOD Conference* (Dallas, Texas, 2000), ACM.

[3] CHEN, K., KAVULURU, R., AND GUO, S. Rasp: Efficient multidimensional range query on attack-resilient encrypted databases. In *ACM Conference on Data and Application Security and Privacy* (2011), pp. 249–260.

[4] CHEN, K., AND LIU, L. Geometric data perturbation for outsourced data mining. *Knowledge and Information Systems 29*, 3 (2011).

[5] FREUND, Y., AND SCHAPIRE, R. E. A short introduction to boosting. In *International Joint Conferences on Artificial Intelligence* (1999), Morgan Kaufmann, pp. 1401–1406.

[6] FUNG, B. C. M., WANG, K., CHEN, R., AND YU, P. S. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Survey 42* (June 2010), 14:1–14:53.

[7] GENTRY, C. Fully homomorphic encryption using ideal lattices. In *Annual ACM Symposium on Theory of Computing* (New York, NY, USA, 2009), ACM, pp. 169–178.

[8] HACIGUMUS, H., IYER, B., LI, C., AND MEHROTRA, S. Executing sql over encrypted data in the database-service-provider model. In *Proceedings of ACM SIGMOD Conference* (2002).

[9] LINDELL, Y., AND PINKAS, B. Privacy preserving data mining. *Journal of Cryptology 15*, 3 (2000), 177–206.

[10] LIU, K., KARGUPTA, H., AND RYAN, J. Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE Transactions on Knowledge and Data Engineering (TKDE) 18*, 1 (2006), 92–106.

[11] NARAYANAN, A., AND SHMATIKOV, V. Robust de-anonymization of large sparse datasets. In *Proceedings of the IEEE Symposium on Security and Privacy* (2008), pp. 111–125.

[12] SCHNEIER, B. Homomorphic encryption breakthrough. http://tinyurl.com/nlzv4w, 2009.

[13] XU, H., GUO, S., AND CHEN, K. Building confidential and efficient query services in the cloud with rasp data perturbation. Accepted by IEEE TKDE in 2012, downloadable from http://arxiv.org/abs/1212.0610, 2012.