Privacy Preserving Aggregation of Secret Classifiers

Gérald Gavin*, Julien Velcin, Philippe Aubertin*****

*ERIC Lab - University of Lyon - 5, avenue Pierre Mendès France

E-mail: gavin@univ-lyon1.fr

**ERIC Lab - University of Lyon - 5, avenue Pierre Mendès France

E-mail: julien.velcin@univ-lyon2.fr

***Axopen, 17, lot colline du Châtel, 01120 Dagneux, France

E-mail: aubertinp@gmail.com

Abstract. In this paper, we address the issue of privacy preserving data-mining. Specifically, we consider a scenario where each member j of T parties has its own private database. The party j builds a private classifier h_j for predicting a binary class variable y. The aim of this paper consists of aggregating these classifiers h_j in order to improve individual predictions. More precisely, the parties wish to compute an efficient linear combination over their classifier in a secure manner.

1 Introduction

We consider a scenario where T parties with private databases wish to cooperate by computing a data-mining algorithm for the union of these databases. Since the databases are all confidential, no party wishes to divulge any content to any other. Let us detail a concrete scenario making our approach particularly relevant where a database is *vertically*[1] partitioned. Each party only knows a subset of explicative variables for all the instances and *the class value y for few instances*. We assume that each party j has inferred a prediction function (classifier) h_j to predict y. These parties could be interested in collaborating to improve each individual prediction performance. A solution would consist of aggregating the different classifiers h_j to build an overall better one h. The simplest way to combine these classifiers consists of predicting the majority class. This approach is naive when, for instance, most of the parties have almost the same low quality classifier. The aim of this paper is to propose a more appropriate approach. The natural evolution of the previous approaches consists of using linear combinations, i.e. weighted votes. Many ways are proposed in the literature to build such combinations. The most famous algorithm is certainly Adaboost. This algorithm builds convex combinations maximizing margins [20]. In [21], the authors propose upper-bounds on the generalization error independent of the combination size.

A key problem that arises in any collection of data is confidentiality. The need for privacy is sometimes due to legal requirements (e.g., medical databases) or can be motivated by

[1]More general scenarii can be imagined where data are both *horizontally and vertically* partitioned: the only constraint is that each pair of parties $(j, k) \in \{1, ..., T\}^2$ can compute respectively $h_j(i)$ and $h_k(i)$ for all instances i belonging to a common subset. This general case is not studied in this paper but it is discussed in section 9.

business interests. In this paper, the parties do not wish to share any information about their private data or their classifier h_j. Efficient cryptographic primitives [6] allow to efficiently evaluate any distributed arithmetical circuit[2]. To use these cryptographic primitives, the choice of the aggregating algorithm should be restricted: it should be essentially "arithmetic", meaning that the only allowed operations are addition and multiplication. This excludes, for instance, the algorithm Adaboost which requires logarithmic and exponential computations. In order to satisfy both machine learning and privacy constraints, we choose to minimize the quadratic error on a convex C. This quadratic error is close to the exponential error minimized by Adaboost. In addition, this optimization can be done with a simple descent gradient algorithm. However, the projection operator is not "arithmetic" if C is the set of the convex combinations. To overcome this issue, the convexity constraint is partially removed, i.e. the positivity constraint on the coefficients α_i is released. In other words, the parties will build a linear combination $h_\alpha = \alpha_1 h_1 + ... + \alpha_T h_T$ minimizing the quadratic error under the constraint $\sum_{j=1}^{T} \alpha_j = 1$.

In Section 3, we propose an algorithm, called QEM (Quadratic Error Minimization), that achieves this optimization process. This algorithm is experimented in Section 4: it is shown that this weighting scheme can be dramatically better than the naive one (equivalent to a simple vote). In Section 7, we present the protocol SMEM (Secure Multi-party Error Minimization) that allows the parties to securely implement this scheme. The cryptographic tools used in this paper are threshold homomorphic encryption schemes [10], [9] allowing any arithmetical circuit to be computed securely [6]. These tools are presented in Section 6. SMEM is shown to be secure against any polynomial adversary. Extensions to the multiclass case are presented in section 8.

2 Problem Statement

In [23, 25, 5] the authors propose two-party protocols to build a decision tree, an SVM or a neural network. In these papers, it is assumed that a database is horizontally or vertically partitioned. Parties then jointly infer a classifier from the whole database. However, intermediate computations are made public, leaking information about private data. For instance, in [23], the entropy computations are public. In [25], the authors assume that parties do not collude, i.e. an adversary controls at most one party. Pinkas and Lindell [16] focus on the problem of decision tree learning with the popular ID3 algorithm. Their protocol is shown secure against passive (semi-honest) adversaries. In [11], authors propose protocols that allow two or more participants to construct a boosting classifier without explicitly sharing their data sets. However, these protocols are not private, even against passive adversaries. In [14], the authors extend the notion of privacy preservation, or secure multi-party computation, to gradient-descent-based techniques. However, the proposed two-party protocols are costly and they are not shown secure against active adversaries. The protocols of this paper are shown to be correct and private against any active polynomial adversary.

Let us concretely precise the setup where the main protocol SMEM of this paper is applicable. Let us suppose that T parties wish collaborating in a prediction problem where the class variable is denoted by y. They first should agree on a list $\mathcal{L}$ of n instances. It can be assumed that each party j **knows an explicative variables tuple** x_{ij} **for each instance** $i \in \mathcal{L}$

[2]More general results (see [17]) state that any function can be securely computed with computational security in presence of a static, active adversary corrupting less than n/2 parties. However the evaluation of the resulting SMC circuit is too expensive in the real-life.

and the class value y_{ij} only for a subset z_j of instances. In this case, the party j can infer a classifier h_j on the training set[3]

$$\{(x_{ij}, y_{ij}) | i \in z_j\}$$

The prediction of the party j on any instance $i \in \mathcal{L}$ is denoted by $h_j(i)$. The Parties wish to build an efficient aggregating classifier without revealing about their private data (including the classifier h_j). In SMEM, parties send encryptions of their prediction $h_j(i)$ and their value y_{ij} (by convention $y_{ij} = 0$ if this value is not known).

First of all the parties should agree on a class value. Indeed, they are not ensured to own the same class value y_{ij} (because of errors or because of missing information). Thus, they should build a common class value ν_i by aggregating the values y_{ij}. The first intuitive idea consists of choosing the most represented value ν_i among the $(y_{ij})_{i=1,...,T}$. SMEM implements this choice but other choices can be also relevant in some applications. Let us consider, for instance, a credit approval case where different banks are combining previous default loans on the same customer base. A customer i may have defaulted on his loan with bank B_1 ($y_{i1} = 1$) and not with other banks $(B_j)_{j=1,...,T}$ ($y_{ij} = -1$ for all $j = 2, ..., T$). In this case, -1 is the most represented class but it could be more relevant to state $\nu_i = 1$ (meaning that there is a problem with the customer i) or to transform this problem in a multi-class problem. SMEM can be straightforwardly adapted to any choice of ν_i (provided it exists efficient cryptographic protocols to compute encryptions of ν_i given encryptions of y_{ij}).

Parties then compute an encryption of the common error m_{jk} (within a multiplicative factor) between each pair (h_j, h_k), i.e.

$$m_{jk} = \frac{4}{n} |\{i \in \{1, ...n\} | h_j(i) = h_k(i) \neq y_i\}|$$

In the next section, it is shown that the quadratic error of a linear combination can be expressed with these values m_{jk}. We propose an arithmetical algorithm QEM minimizing this error (over a well-chosen set C of linear combinations) which is securely implemented in SMEM. More general *scenarii*, where $h_j(i)$ cannot be computed on all the instances of $\mathcal{L}$, could be considered. Indeed, the estimation of the common errors between h_j and h_k only requires that there exists a subset of $\mathcal{L}$ such that $h_j(i)$ and $h_k(i)$ can be computed. It will be discussed in Section 9.

3 Quadratic Error Minimization

Let us suppose that the parties agree on a sample z_n of n instances. Let $y_i \in \{-1, 1\}$ be the class of the instance i and x_{ij} the information (e.g. a predictive variables vector) known by party j about instances i. We also suppose that parties have inferred a classifier h_j in order to predict the class variable y. We will denote by $h_j(i) \in \{-1, 1\}$ the class predicted by the classifier h_j on the instance i. In this section, we propose to build a linear combination over the classifiers h_j minimizing the quadratic error, noting that this error is close to the one minimized by Adaboost, i.e. the exponential error[4]. Minimizing quadratic error under convexity constraints can be done in polynomial time with gradient descent algorithms. However the convexity constraint implies a projection phase which is not "arithmetic". Thus,

[3]In fact, our scheme is more general in the sense that it is only required that each party is able to produce a prediction, via an expert for instance, for each instance $i \in \mathcal{L}$.

[4]The exponential error is equal to $\frac{1}{n} \sum_{i=1}^{n} e^{-h_\alpha(i) y_i}$

the classical cryptographic tools are not adapted for this algorithm. The solution that we propose consists of relaxing this convexity constraint. For concreteness, the positivity constraint is removed and we propose to find a linear combination $h_\alpha = \alpha_1 h_1 + ... + \alpha_T h_T$ minimizing the quadrating error under the constraint $\sum_{j=1}^{T} \alpha_j = 1$. By removing the positivity constraint, $\sum_{i=1}^{T} |\alpha_i|$ can be larger than 1. The performance could be degraded if this sum is too large [1]. To overcome this, we penalize large weight combinations[5] by adding a *regularization term* $\gamma \sum_{i=1}^{T} \alpha_i^2$ to the quadratic error where $\gamma \in \mathbb{R}^+$ is a parameter.

3.1 Quadratic Error

In this section, we denote by H the set of these classifiers, i.e. $H = \{h_j | j = 1, ..., T\}$. I will denote the identity matrix of size T and C will denote the following convex set

$$C = \left\{ \alpha \in \mathbb{R}^T \mid \forall j \in \{1, ..., T\}\ \alpha_j \in \mathbb{R}\,;\ \sum_{j=1}^{T} \alpha_j = 1 \right\}$$

In other words, C is simply the hyperplane $\alpha_1 + ... + \alpha_T = 1$ of $\mathbb{R}^T$. For any $\alpha \in C$, the linear combination $h_\alpha = \sum_{j=1}^{T} \alpha_j h_j$ can be classically transformed into a binary classifier $\bar{h_\alpha}$ by applying the function sign, i.e. $\bar{h_\alpha} = \mathsf{sign}(h_\alpha)$. As discussed previously, this section proposes an algorithm to minimize the quadratic error of h_α denoted $er_H(\alpha)$[6] as defined by,

$$er_H(\alpha) = \frac{1}{n} \sum_{i=1}^{n} (h_\alpha(i) - y_i)^2 + \gamma \sum_{j=1}^{T} \alpha_j^2$$

In the next lemma, it is shown that for any $h \in C$, $er_H(\alpha)$ only depends on the values[7] $(m_{jk})_{(j,k) \in \{1,...,T\}^2}$ defined by

$$m_{jk} = \frac{1}{n} \sum_{i=1}^{n} (h_j(i) - y_i)(h_k(i) - y_i)$$

These coefficients are related to the common error between h_j and h_k, i.e.

$$m_{jk} = \frac{4}{n} |\{i \in \{1, ...n\} | h_j(i) = h_k(i) \neq y_i\}|$$

Let us observe that $m_{jk} \leq m_{jj}$ and $m_{jj} = 4.er_{z_n}(h_j)$ where $er_{z_n}(h_j)$ designates the empirical error of h_j. In the following lemma, M will denote the $T \times T$ matrix of the coefficients m_{jk}, i.e. $M = [m_{jk}]$.

Lemma 1. *Let $\gamma \in \mathbb{R}$ be a regularization parameter. M is symmetric, defined positive and for any $\alpha \in C$,*

$$er_H(\alpha) = \alpha^T (M + \gamma I) \alpha$$

[5]Directly inspired of the weight decay method for the artificial neural networks.

[6]$er_H(\alpha)$ upper-bounds the classification error of $\bar{h_\alpha}$.

[7]The evaluation of m_{jk} only requires that parties j and k have common instances.

Proof. Let $\alpha \in C$. First, let us consider the quantity

$$Q(\alpha) = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{T}\alpha_j h_j(i) - y_i\right)^2$$

By using the fact that $\sum_{j=1}^{T}\alpha_j = 1$ we can state

$$Q(\alpha) = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{T}\alpha_j\left(h_j(i) - y_i\right)\right)^2$$

By developing and by inverting sums, we get

$$Q(\alpha) = \sum_{j=1}^{T}\sum_{k=1}^{T}\alpha_j\alpha_k\left(\frac{1}{n}\sum_{i=1}^{n}\left(h_j(i) - y_i\right)\left(h_k(i) - y_i\right)\right) = \alpha^T M\alpha$$

The result is obtained by noticing that $er_H(\alpha) = Q(\alpha) + \alpha^T(\gamma I)\alpha$
□

□

In the next section, we propose a gradient descent algorithm to minimize the function er_H.

3.2 Algorithm QEM

In this section, we are looking for an algorithm minimizing $er_H(\alpha)$. In next sections, this algorithm will be transformed for observing a secure multi-party protocol. In order to use classical cryptographic primitives, the only allowed computations are arithmetic operators (+ and $\times$). It excludes, for example, algorithms requiring normalization steps. In this section, we consider the algorithm QEM which minimizes $er_H(\alpha)$ over the convex C (defined in the previous section) with a gradient descent algorithm. Let us detail its principle. First, the gradient ∇er_H of er_H should be computed

$$\frac{\partial er_H}{\partial \alpha_j}(\alpha) = 2\left(\sum_{k=1}^{T} m_{jk}\alpha_k + \gamma m_{jj}\alpha_j\right)$$

Thus, $p = (M + \gamma I)\alpha = \nabla er_H(\alpha)/2$. QEM iteratively updates α by computing

$$\alpha^{new} = \mathbb{P}_C\left(\alpha + \frac{\rho}{2}\nabla er_H(\alpha)\right)$$

where $\rho \in \mathbb{Q}$ is a parameter (called sometimes *learning rate* in datamining) and $\mathbb{P}_C$ the projection operator over C. Because the vector $u = (1, ..., 1) \in \mathbb{R}^T$ is orthogonal to C, $\mathbb{P}_C$ simply consists of adding the same value V_α to each component, i.e.

$$\alpha^{new} = \alpha + \frac{\rho}{2}\nabla er_H(\alpha) + V_\alpha u$$

with V_α chosen such that α^{new} belongs to the convex C, here $V_\alpha = \frac{\rho}{T}\sum_{i=1}^{T} p_i$. We notice that all the steps of QEM are *arithmetical* ($\frac{\rho}{T}$ can be pre-computed).

Proposition 2. *Let λ_1 be the greatest eigenvalue of $M + \gamma I$. For any $0 < \rho < \frac{2}{\lambda_1}$, QEM converges to the combination which minimizes the quadratic error over the convex C.*

Proof. (Sketch.) First, let us check that QEM is a projected gradient descent algorithm where the projection is done over C.

Algorithm QEM

```
Inputs: $K \in \mathbb{N}$, $\gamma \in \mathbb{Q}$ and $\rho \in \mathbb{Q}$
    1. $\alpha = (1/T)_{j=1,\ldots,T}$
    2. For $k = 1$ to $K$
        (a) $p = (M + \gamma I)\alpha$
        (b) $\alpha_j = \alpha_j + \frac{\rho}{T}\sum_{i=1}^{T}(p_j - p_i)$
    3. output $\alpha$
```

$\alpha_j + \frac{\rho}{T}\left(\sum_{i=1}^{T}(p_j - p_i)\right)$
$= \alpha_j + \rho p_j + \frac{\rho}{T}\sum_{i=1}^{T} p_i$
$= \alpha_j + \frac{\rho}{2}\nabla er_H(\alpha) + V_\alpha u$ with $V_\alpha = \frac{\rho}{T}\sum_{i=1}^{T} p_i$ and $u = (1, ..., 1) \in \mathbb{Q}^T$

Then, it suffices to prove that V_α is well-chosen by verifying that the updated coefficient vector α belongs to the convex C. Indeed,
$\sum_{j=1}^{T}\left(\alpha_j + \frac{\rho}{T}\left(\sum_{i=1}^{T}(p_j - p_i)\right)\right)$
$= \sum_{j=1}^{T}\left(\alpha_j + \rho p_j - \frac{\rho}{T}\sum_{i=1}^{T} p_i\right)$
$= \sum_{j=1}^{T} \alpha_j$
$= 1$
As $M + \gamma I$ is positive definite, the associated bilinear form is elliptic. Thus, the projected gradient algorithm QEM converges if ρ is small enough, i.e. $0 < \rho < \frac{2}{\lambda_1}$.
□ □

4 Experiments

In this section, we experimentally compare our weighting scheme to other weighting schemes. In all of our experiments, $\gamma = 0$. Let us discuss the choice of the learning rate ρ. During the protocol SMEM, the coefficients m_{jk} are kept secret and parties have only access to their encryptions. ρ can be chosen *a priori* without taking into account of these values (for instance as $\lambda_1 \leq 4T$, we can choose $\rho = 1/2T \leq 2/\lambda_1$). In order to converge faster, it would be better to consider values of ρ as close as possible of $2/\lambda_1$. If the choice of ρ is dependent on the value m_{jk}, it cannot be public (otherwise it would reveal information about M) and consequently SMEM could only deal with encryptions of ρ. It seems difficult to secretly and efficiently compute an encryption of λ_1 with classical cryptographic primitives only given encryptions of m_{jk}. However, as M is a *nonnegative* symmetric matrix, it is well-known that

$$\max_{j=1,\ldots,T}\sum_{k=1}^{T} m_{jk} \geq \lambda_1 \geq \min_{j=1,\ldots,T}\sum_{k=1}^{T} m_{jk}$$

Name	Variables number	Instances number
Ionosphere	34	351
BreastW	31	569
Clean	167	476
CreditG	25	1000
Spambase	57	4608

Figure 1: Description of the databases used in our experiments.

In our experiments, we assumed that $\lambda_1 \approx \frac{1}{T}\sum_{j=1}^{T}\sum_{k=1}^{T} m_{jk}$. This naturally leads to choose

$$\rho = 2T/\sum_{j=1}^{T}\sum_{k=1}^{T} m_{jk}$$

This way to choose ρ can be integrated in SMEM (see remark 1). The tests are made on classical benchmarks used in machine learning (they can be found on the UC Irvine Machine Learning Repository)

In our problem, each party has a partial view of the learning set z_n (it is implicitly assumed that the parties agreed on the list $\mathcal{L}$ of the n instances of z_n). In the following, n designates the number of instances and p designates the number of explicative variables. In our experiments, each party j **only knows a (randomly chosen) subset of p' explicative variables**. Each party j knows the values of these p' variables for all instances but the class value y_i is known **only for a (randomly chosen) subset z_j of n' instances**, i.e. $|z_j| = n'$. By summary, each party j knows the values of p' explicative variables for n' labeled instances (z_j) and $n - n'$ unlabeled instances.

Then each party j builds a decision tree h_j with the classical method C4.5 [19]. The classifier h_j is learnt on the learning set z_j: z_j contains n' instances, each one being described by p' explicative variables and the class value y. We denote by H the set of these classifiers, i.e. $H = \{h_j | j = 1, ..., T\}$. Because of the unlabeled instances of $z_n \setminus z_j$, each party j can compute $h_j(i)$ for all the instances of z_n.

The coefficients m_{jk} (let us recall that m_{jk} represents the common error between the classifiers h_j and h_k) are computed over $\bigcup_{j=1}^{T} z_j$ ($\approx z_n$ the whole training set when $Tn' \gg n$): it is implicitly assumed that the parties start pooling their class values. This will be also done, with privacy preserving, in SMEM.

The parameters of these experiments will be the number of parties T, the number p' of variables and the number n' of class values known by each party. The generalization error is estimated with 10-fold cross-validation. We compare the 4 following methods:

- SV: Simple vote, i.e. the classifiers h_j are uniformly weighted by $1/T$
- QEM[K]: see previous section. K refers to the number of iterations.
- QEMC[K]: This algorithm is a gradient descent algorithm. It minimizes the quadratic error on the set of **convex** combinations over H, i.e $co(H)$. To achieve this, it suffices to adapt the projection operator $\mathbb{P}_C$ of QEM.
- Adaboost[K] : K iterations of the algorithm Adaboost where the weak classifiers set is H. Note that the number of non-zeros coefficients in the output combination is smaller than K.

$n' = n/3$	SV	QEM[10]	QEM[50]	QEMC[50]	Adaboost[10]
Clean	26.77	18.33	18.12	18.12	18.33
Credit	31.10	27.90	29.15	28.00	30.00
SpamBase	14.74	8.38	8.26	8.51	9.44
BreastW	7.46	5.79	6.14	6.32	6.58
Ionosphere	9.58	6.90	7.46	7.61	8.59

$n' = n/5$	SV	QEM[10]	QEM[50]	QEMC[50]	Adaboost[10]
Clean	34.27	24.17	24.01	25.00	24.74
Credit	30.95	28.00	27.95	27.40	29.98
SpamBase	15.47	11.02	9.91	10.09	10.98
BreastW	8.68	6.67	6.54	6.14	6.62
Ionosphere	14.01	10.49	10.14	10.70	11.06

$n' = n/7$	SV	QEM[10]	QEM[50]	QEMC[50]	Adaboost[10]
Clean	34.22	25.73	25.26	25.26	25.47
Credit	30.47	27.38	28.03	27.42	27.68
SpamBase	14.17	9.47	9.57	9.71	10.81
BreastW	7.94	6.18	6.27	6.62	7.24
Ionosphere	13.03	9.93	10.07	10.42	10.85

Figure 2: 10-cross validation error of SV, QEM, QEMC, Adaboost for the parameters $T = 10$; $p' = p/10$; $n' = n/3, n/5, n/7$

Results. In all of our experiments, the generalization error significantly and sometimes drastically decreased with the number of iterations in QEM. At convergence, the error rate is sometimes half of the error rate during the first iteration. Secondly, convergence is reached quickly. After 10 iterations, approximately 50% of the improvements are already done. Figures 2 and 3 summarize the performances of SV, QEM, QEMC and Adaboost. We see that the performance of the Simple Vote is significantly lower than the performance of the three other methods. This justifies the interest in weighting methods. The second experimental conclusion is that QEM is not significantly worse than the other weighting methods Adaboost and QEMC. In fact, QEM seems even better than these methods. In our opinion, it is just because negative weights are permitted in QEM (we do not see other explanation for the comparison between QEMC and QEM). To confirm this, one could run Adaboost by choosing $H \bigcup -H$ as set of weak classifiers (we did not experiment this) and compare performance. Moreover, when the quality of the classifiers h_j decreases, e.g. the size n' of the learning set z_j decreases, improvements provided by QEM with respect to SV become more significative. It clearly appears by comparing error rates mentioned in figure 3.

5 Security Definitions

In this section, we present classical definitions of Security for Multi-party Computation (SMC). For sake of simplicity, technical tools are omitted [16, 17] and we focus on intuitive ideas behind formal existing security definitions.

Basic properties. Let us start by listing natural intuitive properties that a relevant security definition should encapsulate.

$n = n'$	SV	QEM[10]	QEM[50]	QEMC[50]	Adaboost[10]	Adaboost[20]	Adaboost[100]
Clean	11.87	12.08	11.04	11.87	12.92	12.08	9.79
Credit	30.50	24.90	24.20	23.30	28.40	26.90	27.00
SpamBase	7.55	5.48	4.89	4.67	4.94	4.94	4.89
BreastW	5.26	5.35	5.09	6.14	5.79	5.61	5.26
Ionosphere	10.70	10.70	9.86	9.86	9.86	9.30	9.30

$n' = n/5$	SV	QEM[10]	QEM[50]	QEMC[50]	Adaboost[10]	Adaboost[20]	Adaboost[100]
Clean	16.60	16.53	15.21	15.83	19.37	15.76	15.35
Credit	29.43	26.23	25.57	26.30	27.63	27.20	27.20
SpamBase	8.34	6.58	6.04	6.17	8.14	8.12	8.12
BreastW	5.50	5.67	4.97	4.80	5.20	4.74	4.74
Ionosphere	7.89	7.51	6.85	7.89	8.36	7.89	8.26

$n' = n/15$	SV	QEM[10]	QEM[50]	QEMC[50]	Adaboost[10]	Adaboost[20]	Adaboost[100]
Clean	23.68	22.78	21.39	23.26	24.10	21.67	19.93
Credit	29.25	27.00	25.20	25.55	27.40	27.05	26.70
SpamBase	9.93	7.27	6.59	7.05	8.65	8.61	8.57
BreastW	5.35	5.35	4.47	5.09	5.88	5.88	5.79
Ionosphere	11.41	9.58	8.73	10.42	9.86	9.72	10.99

$n' = /50$	SV	QEM[10]	QEM[50]	QEMC[50]	Adaboost[10]	Adaboost[20]	Adaboost[100]
Clean	27.81	26.87	22.29	25.47	26.67	25.42	22.24
Credit	29.65	28.35	24.30	25.08	27.20	26.27	25.60
SpamBase	12.34	7.56	6.70	7.13	7.75	7.85	7.71
BreastW	7.15	6.75	5.13	6.23	7.06	5.66	5.48
Ionosphere	9.15	7.61	6.76	7.89	8.38	7.96	7.89

$n' = n/70$	SV	QEM[10]	QEM[50]	QEMC[50]	Adaboost[10]	Adaboost[20]	Adaboost[100]
Clean	38.15	32.16	25.39	27.07	27.86	25.78	25.91
Credit	29.81	28.00	26.50	26.75	26.50	26.88	26.69
SpamBase	11.59	9.07	7.97	8.40	9.93	9.91	9.91
BreastW	12.85	6.36	5.32	6.45	5.79	5.18	5.09
Ionosphere	38.59	16.76	11.34	13.87	11.48	10.77	11.27

Figure 3: 10-cross validation error of SV, QEM, QEMC, Adaboost for the parameters $T = 100$; $p' = p/10$; $n' = n, n/5, n/15, n/50, n/70$

- *Correctness:* each party is guaranteed that the output it receives is correct (note that it is not guaranteed to receive an output).
- *Privacy:* parties do not learn anything other than the result and what can be inferred from the result and their input.
- *Independence of inputs:* private inputs must be chosen independently by the involved parties. In particular, corrupted parties cannot choose their inputs as a function of honest parties'inputs.

Other natural properties could be considered (e.g. fairness,...). However, a security definition cannot be a list properties to satisfy: nothing would ensure that failures cannot be derived from missing properties.

Adversarial power. Let π be a T-party protocol. We look at the situation where the protocol is executed on an open broadcast network in presence of an polynomial adversary A, i.e. A is allowed to run in probabilistic polynomial-time. It is assumed that there is subset of $t < T$ parties which are corrupted by A. The adversary A is assumed static, meaning that the subset of parties controlled by A remains unchanged during the protocol. It is also assumed active, meaning that the corrupted parties can arbitrarily deviate from the protocol specification, according to the adversary's instructions. In particular, an adversary can replace corrupted parties'inputs with other ones, abort the protocol, send wrong values, etc...

The Ideal/real paradigm. In order to prove security in the Multi-Party Computation (SMC) model, the real execution of a protocol is compared to an ideal one. In the ideal model, we assume the existence of an uncorrupted oracle. Parties send their private inputs to the oracle which correctly and securely computes the outputs and sends them to the participating parties. In the ideal/real paradigm, the security is reached if the ideal model adversary can simulate real executions (complete transcripts of the execution) of the protocol. An important issue for datamining applications consists of noticing that an adversary can replace the corrupted parties' inputs with arbitrary other ones in the ideal model. It implies that an adversary controlling parties during the execution of a secure protocol can also replace the inputs of the corrupted parties. It is an important limitation of this paper and many others dealing with SMC applied to datamining. This will be discussed in section 9.

The $(g_1, ..., g_l)-$hybrid model. In the $(g_1, ..., g_l)-$hybrid model [3] the execution of a protocol π proceeds as in the real-life model, except, that the parties have access to a trusted party for evaluating the $T-$party functions $g_1, ..., g_l$. In [3], an important modular composition theorem was proven. If a protocol π is proven secure in the $(g_1, ..., g_l)-$hybrid model and if there exists UC-secure[8] protocols π_{g_i} for evaluating the functions g_i then the protocol π remains secure if the trusted party is replaced by the execution of the protocol π_i. The use of this composition theorem greatly simplifies proofs of security. Instead of analyzing a large protocol and proving reductions to subprotocols, it suffices to analyze the security of the large protocol in an idealized model.

[8]Here UC stands for universally composable, which denotes that if a protocol is UC-secure according to the formal definition, then it is secure to use in any context (where it would have been secure to use the ideal functionality).

6 Cryptographic Tools

Homomorphic encryption schemes (El Gamal [9], Paillier [18], Boneh [2]...) have been shown relevant for secure multi-party computation. The most famous of them is attributed to Paillier. This encryption scheme is probabilistic, the public key is a k-bit RSA modulus μ chosen at random and an element $g \in Z^*_{\mu^2}$ of order divisible by μ. The plaintext space for this system is Z_μ. In [7], the cryptosystem is generalized to have plaintext space Z_{μ^s} for any s smaller than the factor of μ and g has order divisible by μ^s. To encrypt $a \in Z_{\mu^s}$, one chooses $r \in Z_{\mu^s}$ at random and computes the ciphertext as $E_{pk}(a) = g^a r^{\mu^s} \mod \mu^{s+1}$. The private key sk is the factorization of n, i.e. $\lambda(\mu)$ or equivalent information. This encryption scheme is shown semantically secure[9] under the well-known DCRA assumption [18]. This encryption scheme is additively homomorphic. Indeed the product of two encryptions is an encryption of the sum of the encrypted value, i.e. $E_{pk}(a)E_{pk}(b) \mod \mu^2$ is an encryption of $a+b$. Several threshold versions have been proposed in the literature [10]. In these versions, the public key pk is known by all parties but the private key sk is shared between parties such that the decryption can be done only if at least $t+1$ parties ($t < T$) agree on it. In the following, we say that S is a $(t+1, T)-$threshold encryption scheme if the secret key is shared between T parties such that the decryption requires at least $t+1$ parties. In other words, t (dishonest) parties cannot decrypt. Consequently, t should upper-bound the number of parties controlled by an adversary A. To build the main protocols of this paper, additional protocols Mult, Sign and EncryptBit are needed.

Definition 3. Let S be a $(t+1, T)-$threshold homomorphic encryption scheme semantically secure. The encryption function is denoted by E_{pk} and we assume the existence of UC-secure protocols Mult, Sign and EncryptBit defined by:

1. **Mult.** Given public encryptions a and b, honest parties can securely compute an encryption of ab
2. **Sign.** Let ξ a security parameter. Given a public encryption of x such that $|x| < \mu/2^{\xi+1}$, honest parties can securely compute an encryption of the sign of x.

$$\mathsf{sign}(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases}$$

3. **EncryptBit.** A party j builds an encryption B of a bit $b \in \{0,1\}$. EncryptBit is a $\Sigma-$protocol [13] allowing the party j to prove (to honest parties) that B encrypts a value belonging to $\{0,1\}$ without revealing it.[10]

A version of protocol Mult can be found in [6]. The protocol Sign can be found in [12] and [22]. Concretely, let X be an encryption of $|x| < \lfloor \mu/2^{\xi+1} \rfloor$. Parties compute encryptions of each bit of the binary decomposition of $v = x + \lfloor \mu/2^{\xi+1} \rfloor \in \{0, ..., \lceil \mu/2^{\xi} \rceil\}$ with the protocol BITREP found in [22] itself based on Mult. Then, parties compare v with $\lfloor \mu/2^{\xi+1} \rfloor$ with a comparison protocol found in [12]. A version of EncryptBit can be found in [7]. The communication cost and the time complexity of these protocols are linear with respect to the parties numbers, i.e. $O(T)$. **The security of all the protocols in this paper will be proven secure in the (Mult, Sign, EncryptBit)-hybrid model.**

[9]Encryptions cannot be distinguished from random values in polynomial time.

[10]At the end of the execution of EncryptBit, each honest party $i \neq j$ is convinced that B encrypts a value $b \in \{0,1\}$ without learning b, i.e. if b is randomly chosen then the party i cannot guess b with a probability significantly better than 1/2.

If a party fails to complete a step during any of the (sub)protocols (e.g., if a proof fails), then that party is simply discarded and the (sub)protocol is rerun by the remaining parties. We will not describe this explicitly for the protocols in this paper. For $0 \leq t < n/2$, the case of a dishonest minority, the above protocols can be directly applied. For $n/2 \leq t < n$, the case of a dishonest majority, adaptations are required to achieve various degrees of fairness.

Notation. *Let X, Y be encryptions of x, y, i.e. $X = E_{pk}(x)$ and $Y = E_{pk}(y)$. $X \oplus Y$ and $X \ominus Y$ will denote an encryption of $x + y$ and $x - y$. These encryptions can be obtained by using the homomorphic properties of the encryption scheme S. In the same way, $X \otimes Y$ will denote an encryption of xy. This encryption is obtained by invoking* Mult*.*

7 The Protocols

In this section, we assume that the parties have jointly generated a pairwise (pk, sk) by invoking the function Generation of a $(t+1, T)-$threshold homomorphic encryption scheme S satisfying the definition 1. The domain of E_{pk} is assumed to be the ring $\mathbb{Z}_\mu$. Let us assume the existence of an adversary A controlling less than t parties. A knows private data of all controlled parties and it can replace any controlled party in the protocol. Here, adversaries can be active, meaning that they can deviate from the protocol in any way [17]. In this section, the protocols are shown secure against any polynomial adversary: it implies that they are correct and private against any adversary. Intuitively, a protocol is said to be correct if an honest party outputs the correct value (or does not output anything if the protocols fails). A protocol is said to be private against an adversary A if A cannot learn anything about the private data of an honest party (except what it can learn with the output and its own private data). In our case, A will not learn anything about the classifiers h_j and the class values y_{ij} of the honest parties. In particular, A will not learn the coefficients m_{jk} and the coefficients α_j output by QEM.

7.1 Protocol SMEM

Informally, SMEM aims to securely compute encryptions Δ_j of the coefficients α_j output by QEM. More precisely, SMEM outputs encryptions Δ_j of $C_K \alpha_j \mod \mu$ with $C_K \in \mathbb{N}$.

A pre-prepossessing task consists of computing encryptions M_{jk} of the coefficients m_{jk}. Let us suppose that each party j has a private database D_j containing a binary variable y. Each party j has built a private classifier h_j to predict a class variable y. Here, it is assumed that parties agree on an instances list[11]. Given an instance $i \in \{1, ..., n\}$, $h_j(i)$ denotes the predicted class of h_j and y_{ij} denotes the class value of the instance i in the private database D_j. By convention, $y_{ij} = 0$ if this value is missing in D_j. At the beginning of the protocol SMEM, each party j broadcasts encryptions H_{ij} and Y_{ij} respectively $h_j(i)$ and y_{ij} for each instance $i = 1, ..., n$. Note that to prove that $h_j(i) \in \{-1, 1\}$ and $y_{ij} \in \{-1, 0, 1\}$, it suffices to check that $(h_j(i) + 1)/2$ and y_{ij}^2 belong to $\{0, 1\}$. This will be used in step 1 of SMEM.

As discussed in Section 2, y_{ij} can differ from the true class value y_i (if it exists), i.e. $y_{ij} \neq y_i$. For these reasons, on each instance i, the parties must agree on a class value ν_i computed with respect to the values y_{ij}[12]. We propose to define ν_i as the class the most represented among the values $(y_{ij})_{j=1,...,T}$, i.e. $\nu_i = \mathsf{sign}\left(\sum_{j=1}^{T} y_{ij}\right)$. Note that $\nu_i = 0$ if the classes are equally represented or if this value is unknown by all the parties, i.e. $y_{ij} = 0$ for all

[11]The method for this is not explained here.

[12]Note that ν_i could differ from y_i.

$j = 1, ..., T$: in this case, the instance i is discarded. Other aggregating choices could be considered (see section 2). SMEM starts by computing an encryption Y_i of ν_i. It is done in step 3 of SMEM by using homomorphic properties and by invoking the protocol Sign.

Let E be the set of instances for which the class ν_i is defined, i.e. $E = \{i \mid \nu_i \neq 0\}$. By noticing that $|E| = \sum_{i=1}^{n} \nu_i^2$, an encryption U of $|E|$ is computed in the step 3. In the following of the protocol, the instances $i \notin E$ are implicitly discarded. Parties then must securely compute encryptions of the coefficients m_{jk} (these errors are computed over E). More precisely, parties compute encryptions M_{jk} of the integers $|E|m_{jk}$ defined by

$$|E|m_{jk} = \sum_{i \in E}(h_j(i) - \nu_i)(h_k(i) - \nu_i) = \sum_{i=1}^{n} \nu_i^2 (h_j(i) - \nu_i)(h_k(i) - \nu_i)$$

In QEM, each output α_j can be written as an arithmetical expression of the inputs. Step 6 of SMEM is simply the transposition of QEM in the *encrypted world*. The main difficulty is that QEM manipulates rational numbers while our cryptographic primitives are defined on the finite ring Z_μ. To overcome this, we consider integers a and v such that $\tilde{\gamma} = \gamma a$ and $\tilde{\rho} = v\rho/T$ are integers. At each iteration k in the loop of step 6, instead of computing an encryption $\Delta_j^{[k]}$ of $\alpha_j^{[k]}$ (where $(\alpha_j^{[k]})_{j=1,...,T}$ is the tuple computed at the k^{th} iteration in QEM), $\Delta_j^{[k]}$ encrypts $\alpha_j^{'[k]} = T(a|E|v)^k \alpha_j^{[k]} \mod \mu$.

Remark 4. The learning rate ρ is chosen *a priori* by parties. We do not explicit this choice. For instance, they can jointly choose $\rho = 1/2T \leq 2/\lambda_1$. This choice does not reveal any information about the coefficients m_{jk} but larger values of ρ (close to $2/\lambda_1$) would reduce the number of iterations K required to reach the convergence. In our experiments (presented in section 4), $\rho = 2T/\sum_{j=1}^{T}\sum_{k=1}^{T} m_{jk}$. This choice can be integrated in SMEM. Indeed, it suffices to state $\tilde{\rho} = 2$ and $V = E_{pk}\left(\sum_{j=1}^{T}\sum_{i=k}^{T} m_{jk}\right)$ where V can be computed after step 4 by simply using homomorphic properties.

Remark 5. The regularization parameter γ is chosen *a priori* by parties. This parameter should be adjusted in order to resist against overfitting. Here, we do not precise how this choice is done.

Proposition 6. *Let $(\alpha)_{j=1,...,T}$ be the vector output by QEM. Assume the $(t+1, T)-$threshold homomorphic encryption scheme S semantically secure. SMEM is secure, in the (Mult, Sign, EncryptBit)-hybrid model, against any polynomial adversary controlling less than t parties. Moreover, SMEM outputs encryptions Δ_j of $\alpha'_j = C_K \alpha_j \mod \mu$ with $C_K < T(avn)^K$.*

Proof. Assuming correctness of the protocols Mult, Sign and EncryptBit, it is easy to check that the five first steps are correct (we just have to notice that $S_i = \bigoplus Y_{ij}$ encrypts a value smaller than $T < \mu/2^\xi$ ($\mu > 2^{1024} \gg T$)). At the beginning of step 6, Δ_j encrypts 1 (T times the initial coefficients α_j of QEM), U encrypts $|E|$, M_{jk} encrypts $|E|m_{jk}$ for $(j,k) \in \{1, ..., T\}^2$,

For any $k \in \mathbb{N}$, let us denote by $\alpha_j^{'[k]}$ the encrypted value by Δ_j at the end of the k^{th} iteration in the loop of step 6. Let $k \in \mathbb{N}^*$. Let us assume that it exists a constant C_k such that $\alpha_j^{'[k]} = C_k \alpha_j^{[k]}$ where $\alpha_j^{[k]}$ is the current value of α_j in QEM at the end of the step k. Because $\tilde{\gamma} = a\gamma$, $P_j^{[k+1]}$ (the current encryption P_j at the $(k+1)^{th}$ iteration) encrypts a

Protocol SMEM

Let $K \in \mathbb{N}$, $\gamma \in \mathbb{Q}$, $\rho \in \mathbb{Q}$ be parameters chosen a priori. *Let $a, v \in \mathbb{N}$ be integers such that $\widetilde{\gamma} = \gamma a \in \mathbb{N}$ and $\widetilde{\rho} = v\rho/T \in \mathbb{N}$. Let A, V, Γ, Υ be encryptions of $a, v, \widetilde{\gamma}, \widetilde{\rho}$ Each party has computed a classifier h_j. For each $i = 1, ..., n$, each party $j = 1, ..., T$ computes and broadcasts encryptions Y_{ij}, H_{ij} of $y_{ij}, h_j(i)$.*

Public inputs: K, Γ, Υ, A, V, Y_{ij}, H_{ij}, $(i, j) \in \{1, ..., n\} \times \{1, ..., T\}$.

```
1. Each party $j$ prove that $Y_{ij}$ and $H_{ij}$ encrypt a value belonging respectively to $\{-1, 1\}$
   and $\{-1, 0, 1\}$ by invoking EncryptBit for all $i = 1...n$ on respectively $(H_{ij} \oplus E_{pk}(1)) \otimes E_{pk}(2^{-1})$ and $Y_{ij} \otimes Y_{ij}$.
2. $U = E_{pk}(0)$
3. for $i = 1$ to $n$
   (a) $S_i = \bigoplus_{j=1}^{T} Y_{ij}$
   (b) $Y_i = \mathsf{Sign}(S_i) \ominus \mathsf{Sign}(E_{pk}(-1) \otimes S_i)$
   (c) $U = U \oplus Y_i \otimes Y_i$
4. compute for $(j, k) \in \{1, ..., T\}^2$
   $$M_{jk} = \bigoplus_{i=1}^{n} Y_i \otimes Y_i \otimes (H_{ij} \ominus Y_i) \otimes (H_{ik} \ominus Y_i)$$
5. $\Delta_j = E_{pk}(1)$ for all $j = 1, ..., T$
6. for $k = 1$ To $K$
   (a) $P_j = A \otimes \left(\bigoplus_{j'=1}^{T} M_{jj'} \otimes \Delta_{j'}\right) \oplus (\Gamma \otimes M_{jj} \otimes \Delta_j)$ for all $j = 1...T$
   (b) $\Delta_j = (V \otimes A \otimes U \otimes \Delta_j) \oplus \left(\bigoplus_{k=1}^{T} (P_j \ominus P_k)\right) \otimes \Upsilon$ for all $j = 1...T$
7. output $\Delta_j$ for $j = 1, ..., T$
```

value $p_j^{'[k+1]}$ equal to

$$p_j^{'[k+1]} = a|E| \left(\sum_{i=1}^{T} m_{ji} \alpha_k^{'[k]} + \gamma m_{jj} \alpha_j^{'[k]} \right) = C_k a|E| p_j^{[k+1]}$$

where $p_j^{[k+1]}$ is the value of p_j in QEM at the $(k+1)^{th}$ iteration. Then

$$\alpha_j^{'[k+1]} = C_k \left(av|E| \alpha_j^{[k]} + av|E| \frac{\rho}{T} \sum_{i=1}^{T} \left(p_j^{[k]} - p_i^{[k]} \right) \right) = C_k av|E| \alpha_j^{[k+1]}$$

It follows that

$$C_{k+1} = av|E| C_k$$

As $C_0 = T$ (Δ_j initially encrypts $T\alpha_j^{[0]}$), we show by induction that SMEM outputs encryptions of

$$T(av|E|)^K \alpha_j^{[K]} \mod \mu$$

It proves correctness. To argue that no information on the private values y_{ij} and $h_j(i)$, let us slightly modify the protocol allowing parties to output all public encryptions (inputs and intermediate computed encryptions). Let us denote this new protocol as SMEMm. Assuming that the encryption scheme S is semantically secure, the honest parties cannot deduce anything more about the encrypted value by Δ_j (the corrupted parties can output these encryptions in SMEM anyway). Thus, it suffices to prove security of SMEMm

Let us build a polynomial simulator S which produces a simulated transcript statistically, indistinguishable from the complete transcript of the execution of SMEMm. This simulator is given as input the start-state of the adversary A (in particular it is assumed to know the auxiliary input z), the public key pk, the encryptions input by the honest parties and the expected output of the honest parties (i.e. all the encryptions computed during by SMEM) but of course the simulator does not know private data and private decryption shares of the honest parties. In the (Mult, Sign, EncryptBit)-hybrid model, the (polynomial) simulators of Mult, Sign, EncryptBit can be used by S. These simulators should input the encryptions input and output by the honest parties in the real-life execution[13]. These encryptions are known by the simulators because all the intermediate encryptions are assumed known by S. Thus, each real-life execution (in SMEMm) of Mult or Sign can be achieved by S by invoking the simulators of Mult and Sign. It implies that SMEMm itself can be (polynomially) simulated.

□ □

7.2 Protocol Prediction

In this section, it is assumed that parties have already executed SMEM and they have encryptions Δ_j of $C_K \alpha_j \mod \mu$ with $C_K \leq T(avn)^K$, for all $j = 1, ..., T$ (see previous section). Let x be a new instance. To securely compute the sign of $\alpha_1 h_1(x) + ... + \alpha_T h_T(x)$, each party j computes $h_j(x)$ and broadcasts its encryption Y_j. Then parties jointly compute an encryption C of $C_K.(\alpha_1 h_1(x) + ... + \alpha_T h_T(x)) \mod \mu$ by applying Mult. Then, by invoking the protocol Sign, the parties get an encryption of the predicted class. Finally, the parties can decide to decrypt it or to input it into other protocols. Nevertheless, correctness

[13]EncryptBit is a Σ-protocol. Its simulator only inputs the encryption input in the real-life execution [6].

Protocol Prediction

SMEM(K, Γ, Υ, A, V, Y_{ij}, H_{ij}, $(i,j) \in \{1,...,n\} \times \{1,...,T\}$) should have been invoked: parties have output an encryption Δ_j of $C_K\alpha_j$. Let x be a new instance. Each party j computes $y_j = h_j(x)$.

Public inputs: *encryptions $Y_j = E_{pk}(y_j)$ and Δ_j for $j = 1,...,T$,*

1. each party j proves that Y_j encrypts a value belonging to $\{-1, 0, 1\}$ by invoking EncryptBit on $Y_{ij} \otimes Y_{ij}$
2. compute $C = \bigoplus_{j=1}^{T} \Delta_j \otimes Y_j$
3. **output** $Y =$Sign(C)

is ensured only if μ is large enough. According to the specifications of the protocol Sign (see definition 1), Sign(C) is correct if

$$\mu > 2^{\xi+1}|C_K.(\alpha_1 h_1(x) + ... + \alpha_T h_T(x))|$$

Proposition 7. *Let $\xi \in \mathbb{N}$ be a security parameter and N_K be the Manhattan norm ($\|.\|_1$) of the vector α output by QEM. Assume the $(t+1,T)-$threshold homomorphic encryption scheme S semantically secure and $\mu > 2^{\xi+1}T(avn)^K N_K$, Prediction is secure, in the (Mult, Sign, EncryptBit)-hybrid model, against any polynomial adversary controlling less than t parties.*

Remark 8. In all our experiments, $N_K \approx 1$ meaning that the absolute values of the negative coefficients are small.

Proof. As EncryptBit and Mult are correct steps 1 and 2 are correct. To prove correctness of step 3, it suffices to see that absolute value of the encrypted value by C is smaller than $T(avn)^K N_K$. As $\mu > 2^{\xi+1}T(avn)^K N_K$, the protocol Sign is secure (see definition 1). Thus, the output is correct.

The security proof in the (Mult,Sign,EncryptBit)-hybrid model exactly follows the security proof of proposition 2.

□ □

Thus, by assuming $N_K \approx 1$ (according to remark 3), in order to ensure correctness of SMEM+Prediction it suffices that $\mu \succeq 2^{\xi+1}T(avn)^K$.

7.3 Complexity

The number of encryptions/decryptions/modular exponentiations is linear with respect to T in the protocols Sign, EncryptBit and Mult. The number of invocations of Sign, EncryptBit and Mult is equal to $O(nT^2 + KT)$. Thus, the number of encryptions/decryptions/modular exponentiations computations is equal to

$$O(nT^3 + KT^2)$$

Furthermore, we noted in the previous section that the domain size $\lceil \log_2 \mu \rceil$ is linear (neglecting logarithm factors) in K ($\log \mu = O(\log T + \xi + K \log w'))$. As the complexities of the encryptions/decryptions/modular exponentiations are in $O(\log^3 \mu)$ for classical homomorphic encryption schemes (Paillier, El Gamal), the complexity of SMEM is

$$O\left(K^3 T^2 (nT + K)\right)$$

The protocol Mult is invoked T times by Prediction. Thus, the number of encryptions/decryptions/modular exponentiations is quadratic with respect to T in Prediction. By taking the domain size into account, the complexity is

$$O\left(K^3 T^2\right)$$

These computational costs are not relevant for many real applications. Truncations would be interesting in order to reduce domain size expansion ($\lceil \log \mu \rceil$). In [15], the authors propose a secure multi-party protocol to compute $x \mod a$ and x/a (integer division) given a public divisor a and an encryption of x. This protocol could be used to compute truncations.

8 Multiclass Extension

The main protocol of this paper intrinsically deals with binary class variable. Our protocol cannot be directly extended to multi-class learning problems. Dietterich et Bakiri [8] have proposed an elegant methodology to transform a multiclass problem in several bi-class problems. The principle consists of affecting to each class $u \in \{1, ..., M\}$ with a binary code

$$c_u = (c_u(1), ..., c_u(l))$$

of length $l = O(M)$ where M is the number of classes. The process of building this code is detailed in [8] (it is inspired by corrector codes). Thus, l learning processes are done. In the v^{th} learning process the classes c_u such that $c_u(v) = 1$ are regrouped. We denote the binary classifier generated in the v^{th} learning process by h_v. For an explicative variables vector x, we denote as e_x the $l-$bit vector

$$e_x = (h_1(x), ..., h_l(x))$$

The predicted class u_x is the class whose the code is nearest to e_x with respect to the Hamming distance, i.e.

$$u_x = \underset{u=1,...M}{\arg\min} \left(\sum_{v=1}^{l} |c_u(v) - e_x(v)| \right) = \underset{u=1,...M}{\arg\min} \left(\|c_u - e_x\|_1 \right)$$

In practice, this algorithm has good performance. Minor modifications to our scheme are needed in order to integrate this. Let us roughly outlines these modifications. First of all, the **parties agree on a (public) class code**. In MultiSMEM, each party j sends $2 \times l \times n$ encryptions of $h_{vj}(i)$ and y_{vji} where $h_{vj}(i) \in \{-1, 1\}$ is the prediction of the v^{th} classifier h_{vj} of the party j on the instance i. If the party j believes (knows) that class k of this instance j satisfies $c_k(v) = 1$ (resp. $c_k(v) = -1$) and $y_{vji} = 1$ (resp. $y_{vji} = -1$) and $y_{vji} = 0$ if party j does not have this information. They then build l combinations $\alpha_{v1} h_{v1} + ... + \alpha_{vT} h_{vT}$ for $v = 1, ..., l$ by invoking l times SMEM. They get the $l \times T$ encryptions Δ_{vj} of α_{vj}.

In MultiPrediction, more significant modifications are required. Indeed, parties jointly compute an encryption of each component of the vector e_x in the same way as done in Prediction. However they should then compute an encryption of

$$\arg\min_{u=1,\dots M} (\|c_u - e_x\|_1)$$

As the components of the vectors $c_u(v)$ and e_x are binary,

$$\|c_u - e_x\|_1 = \|c_u - e_x\|_2 = \sum_{v=1}^{l} (c_u(v) - e_x(v))^2$$

Thus, computing encryptions D_u of $\|c_u - e_x\|_1$ is done in step 3 by using the homomorphic properties of the underlined homomorphic encryption scheme (e.g. Paillier) and the protocol Mult. The index u_x such that $u_x = \text{argnin}_{u=1,\dots M} (\|c_u - e_x\|_1)$ can also be found by using only addition and multiplication and the function Sign. Let $z_1, ..., z_{k-1}$ be $k-1$ values such that $z_m = \min_{u=1,\dots,k-1}(z_u)$ for $m \in \{1, ..., k-1\}$ and z_k a new value. It can be seen that

$$\operatorname*{argmin}_{u=1,\dots,k} (z_u) = Sign(z_k - z_m)m + (1 - Sign(z_k - z_m))k$$

This simple idea is used in step 6 to compute an encryption of u_x. At each iteration of the loop, Min encrypts z_m, and I encrypts $\text{argmin}_{u=1,\dots,k} (z_u)$.

Proposition 9. *Assume the $(t+1, T)-$threshold homomorphic encryption scheme S semantically secure and μ sufficiently large. The protocol MultiPrediction is secure, in the (Mult, Sign, EncryptBit)-hybrid model, against any polynomial adversary controlling less than t parties. The complexity of this protocol is*

$$O(M(M+T))$$

9 Discussion and Future Work

SMEM and Prediction were shown secure against any polynomial adversary. However a "malicious" adversary A can always alter its inputs. It can choose "malicious" inputs in order to get relatively large weights α_j for the parties it controls. To get relatively large weights, A is interested in choosing a classifier h_j with a low empirical error (even if h_j has a very bad generalization error). In other words, if it knows the class value for a majority of instances, it could input a classifier with a low empirical error. Thus our scheme is not robust against an adversary which knows a large number of class values.

Furthermore, to predict the class of a new instance x, each party j should input an encryption of $h_j(x)$ into Prediction. An adversary could input a wrong value in order to change the prediction of the aggregated classifier.

To overcome this issue, the classifiers h_j should belong to a restricted class whose VC-dimension [24] should be adapted to the number of instances. For instance, parties could only propose decision trees or neural networks of size lower than a given size. To ensure this, classifiers h_j should be committed such that any party can compute an encryption of the predicted class. However, in this case, information about individual classifiers would be leaked. Another way to proceed would consist of imposing statistical controls on inputs. It should be also interesting to consider the regret minimization setting of online learning where several experts can be combined to make an aggregate prediction in spite of adversaries [4].

Protocol MultiPrediction

MultiSMEM($K, \Gamma, \Upsilon, (Y_{vji}, H_{vji})_{(v,j,i)\in\{1,...,l\}\times\{1,...,T\}\times\{1,...,n\}}$) should have been invoked: parties have output encryptions Δ_{vj} of $C_K\alpha_{vj}$. Let x be a new instance. Each party j computes $y_{vj} = h_{vj}(x)$.

***Public inputs:** encryptions $Y_{vj} = E_{pk}(y_{vj})$ and Δ_{vj} for $j = 1, ..., T$, and $v = 1, ..., l$*

```
1. Each party $j$ proves that $Y_{vj}$ encrypts a value belonging to $\{-1, 0, 1\}$ by invoking EncryptBit on $Y_{ij} \otimes Y_{ij}$.

2. $E_x(v) =$Sign$\left(\bigoplus_{j=1}^{T} \Delta_{vj} \otimes Y_{vj}\right)$ for all $v = 1, ..., l$

3. for $u = 1$ to $M$
   (a) $D_u = E_{pk}(0)$
   (b) for $v = 1$ to $l$
           $D_u = D_u \oplus (E_x(v) \ominus E_{pk}(c_u(v))) \otimes (E_x(v) \ominus E_{pk}(c_u(v)))$

4. $I = E_{pk}(1)$

5. $Min = D_1$

6. for $u = 2$ to $M$
   (a) $S =$ Sign $(Min \ominus D_u)$
   (b) $Min = S \otimes Min \oplus (E_{pk}(1) \ominus S) \otimes D_u$
   (c) $I = S \otimes I \oplus (E_{pk}(1) \ominus S) \otimes E_{pk}(u)$

7. output $I$
```

Finally, it would be interesting to consider more general applications where the database is both *horizontally and vertically* partitioned. In this case, $h_j(i)$ cannot be computed over all the instances. However, the evaluation of the coefficients m_{jk} only requires that the parties j and k knows a common subset of instances such that $h_j(i)$ and $h_k(i)$ can be computed for all the instances i of this subset. If all these subsets have the same size, the adaptation of SMEM is straightforward. How to modify SMEM in the other cases?

Acknowledgements. The authors thank the reviewers for their helpful comments.

References

[1] Peter L. Bartlett. For valid generalization the size of the weights is more important than the size of the network. In *NIPS*, pages 134–140, 1996.

[2] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *TCC*, pages 325–341, 2005.

[3] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.

[4] Nicolò Cesa-Bianchi, Gábor Lugosi, and Gilles Stoltz. Minimizing regret with label efficient prediction. *IEEE Transactions on Information Theory*, 51(6):2152–2162, 2005.

[5] Tingting Chen and Sheng Zhong. Privacy-preserving backpropagation neural network learning. *Trans. Neur. Netw.*, 20(10):1554–1564, 2009.

[6] R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *EUROCRYPT*, pages 280–299, 2001.

[7] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Public Key Cryptography*, pages 119–136, 2001.

[8] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *CoRR*, cs.AI/9501101, 1995.

[9] T. Elgamal. A public key cryptosystem and a signature sheme based on discrete logarithms. In *IEEE transactions on Information Theory*, pages 31:469–472, 1985.

[10] P. Fouque and J. Stern. Fully distributed threshold rsa under standard assumptions. In *IACR Cryptology ePrint Archive: Report 2001/2008*, February 2001.

[11] Sébastien Gambs, Balázs Kégl, and Esma Aïmeur. Privacy-preserving boosting. *Data Min. Knowl. Discov.*, 14(1):131–170, 2007.

[12] Juan A. Garay, Berry Schoenmakers, and José Villegas. Practical and secure solutions for integer comparison. In *Public Key Cryptography*, pages 330–342, 2007.

[13] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *STOC*, pages 291–304, 1985.

[14] Shuguo Han, Wee Keong Ng, Li Wan, and Vincent C. S. Lee. Privacy-preserving gradient-descent methods. *IEEE Trans. Knowl. Data Eng.*, 22(6):884–899, 2010.

[15] J.Guajardo J, B.Mennink, and B.Schoenmakers. Modulo reduction for paillier encryptions and application to secure statistical analysis. In *Financial Cryptography and Data Security - 14th International Conference, FC 2010, Lecture Notes in Computer Science, Springer-Verlag, 8 pages*, 2010.

[16] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In *CRYPTO*, pages 36–54, 2000.

[17] O.Goldreich, S.Michali, and A.Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

[18] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.

[19] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[20] Robert E. Schapire. Theoretical views of boosting. In *EuroCOLT*, pages 1–10, 1999.

[21] Robert E. Schapire, Yoav Freund, Peter Barlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *ICML*, pages 322–330, 1997.

[22] Berry Schoenmakers and Pim Tuyls. Efficient binary conversion for paillier encrypted values. In *EUROCRYPT*, pages 522–537, 2006.

[23] Jaideep Vaidya, Chris Clifton, Murat Kantarcioglu, and A. Scott Patterson. Privacy-preserving decision trees over vertically partitioned data. *TKDD*, 2(3), 2008.

[24] Vladimir Vapnik. Principles of risk minimization for learning theory. In *NIPS*, pages 831–838, 1991.

[25] Hwanjo Yu, Jaideep Vaidya, and Xiaoqian Jiang. Privacy-preserving svm classification on vertically partitioned data. In *PAKDD*, pages 647–656, 2006.