Privacy-preserving boosting

Esma Aïmeur, Gilles Brassard, Sébastien Gambs and Balázs Kégl
Département d'Informatique et Recherche Opérationnelle
Université de Montréal
Montréal, QC Canada
{aimeur,brassard,gambsseb,kegl}@iro.umontreal.ca

Abstract

How to realize an algorithm of the boosting family when the dataset is split between two participants, who although willing to collaborate together for the accomplishment of a task of mutual benefit, wish at the same time to preserve the privacy of their data? We propose MABoost (Multiparty Abstention Boost), a distributed and privacy-preserving boosting by abstention algorithm, as a potential answer to this question.

1 Introduction

Data mining is a rising domain located at the crossroads of databases, statistics and artificial intelligence. Its main goal can be resume as *finding useful information inside a vast amount of data*. At the Age of Information where huge databases are more and more common, it has become very popular and is now used in a wide range of different areas such as finances, sociology or astrophysics, just to name a few.

Secure multiparty computation is the branch of cryptography which deals with *the realization of distributed tasks in a secure manner*, where the definition of security can have different flavors depending of the setting considered such as preserving the privacy of the data and protecting the computation against malicious participants. Typically, a secure multiparty computation is defined as computing a function $f(x, y)$ where the entry x is in the hands of one participant and the entry y is in the hands of the other participant. For the computation to be considered totally secure, the two participants must have learned nothing after the realization of the task other than the output of the function itself and the *a priori* information they had regarding the entries x and y. [Yao 86] was the first to describe a technique which enables the implementation of any probabilistic computation between two participants in a secure manner. Later his results were generalized to the setting of multiple participants [Chaum *et al* 88]. It must be stated, however, that although universal and general, these methods can be very inefficient and heavy in terms of communication complexity when the entries are large and when the function to compute is relatively complex to describe. When this occurs, it is often more desirable to rely on an *ad hoc* solution.

In this paper, we will look at the encounter of these two domains, data mining and secure multiparty computation, in the case where the task is *the construction of a classifier*. In machine learning, *classification* can be defined as the task of *accurately predicting the class of an object from some observations about this object*. Now consider the following scenario:

Alice and Bob are directors of two concurrent banks but they wish to collaborate together in order to perform a task of mutual interest. For example, they might want to construct an apparatus that is able to give an advice on whether or not a person is a good candidate for a loan. Each of the two participants owns a database which contains all the confidential data regarding their clients upon which the classifier has to be built. Clearly, if Alice and Bob could put together their data (e.g. if Bob could send his database directly to Alice), then the problem of building a classifier would be greatly simplified as no distributed or security aspect would have to be taken into account. Unfortunately, this is not the situation as Alice and Bob would like to protect as much as possible the sensitive information they possess regarding their clients, even if they are willing to cooperate. Therefore they wish to build a classifier based on their respective databases in a distributed but also confidential manner, meaning that they want to disclose as little information as possible regarding their respective clients. Notice that, obviously, the final classifier does leak some information about the clients, as in the example given above where we want to distinguish between a good payer and a bad payer, but for the protocol to be considered secure we only require that it does not reveal more than it is possible to learn by looking directly at the description of the final classifier. Comparing different types of classifiers, how well and in which sense they preserve privacy [Evfimievski *et al* 03], is also a very relevant and interesting (and of course complex!) question which we will not discuss in this paper.

In this paper we propose a *boosting*-like algorithm to create the final classifier. We will also constrain ourselves to a security model where the participants are *semi-honest*, also called sometimes *passive* or *honest-but-curious*, meaning that they *follow the execution of protocol without any attempt of cheating but on the other hand they try to learn as much information as possible regarding the other participant's data by looking at the bits exchanged during the protocol* [Goldreich 04]. This model is weaker than the one where we allow some participants to be malicious and to cheat actively during the protocol, but it is however a realistic model that is usually considered in the domain of privacy-preserving data mining. Of course once a protocol is proved secure in this model, it is always possible to attempt to upgrade its security towards a stronger model.

The outline of this paper is the following: first we will describe privacy-preserving data mining and review the three approaches that cope with this problem in the Section 2. We will briefly present the principles of boosting in the Section 3. In Section 4, we will define MABoost (Multiparty Abstention Boost), a boosting algorithm with abstention working in a distributed and secure manner. The complexity of the algorithm (both communication and algorithmic) and its security will be studied respectively in the Sections 5 and 6. Finally we give some possible generalizations to this algorithm in the Section 7.

2 Privacy-preserving data mining

Today, Internet makes it possible to reach and connect sources of information throughout the world, but at the same time raises a lot of questions regarding the security and the privacy of these sources of information. *Privacy-preserving data mining* is an emerging field that studies *how data-mining algorithms can affect the privacy of the data* and tries to *find and analyze new algorithms that will respect and preserve this privacy*. A review of the problematic and the state-of-the-art of privacy-preserving data mining can be found in [Verykios *et al* 04] where some of the algorithms are reviewed and the three different trends of approaches for dealing with privacy-preserving issues in data mining are identified.

The algorithms belonging to the first approach are usually based on *heuristics that modify the values of selected attributes on individual records* in order to preserve privacy. This technique of selective data modification is called *sanitization* and has been proved to be NP-hard [Atallah *et al* 99]. When making data sanitization, one always needs to find an *equilibrium between the amount of privacy and the utility loss* resulting from this sanitiza-

tion process. In order to sanitize data, it is possible to make different modifications on the data such as *altering* the value of an attribute by either *perturbing* it or *replacing* it with the "unknown" value, *swapping* the values of attributes between individual records, *use a coarser granularity* by merging several possible values, use a *sampling* strategy, etc.

With the second approach, the data is modified again but in a *global* rather than a local manner. By *adding noise drawn from a known distribution* (e.g. gaussian or uniform) it is possible to construct a classifier and to apply a *reconstruction algorithm* which will try to clean up the effect of the noise and construct a classifier which is as close as possible to the one constructed on the original distribution [Agrawal and Ramakrishnan 00]. Here also there will be a trade-off between the *level of privacy* (the noise added) and the *quality of the reconstruction* which directly affects the performance of the classifier. This approach can be extended in a natural way to the problem of *density estimation*, where we are trying to estimate directly the shape of the original distribution rather that building a classifier based on this distribution. In [Agrawal and Aggarwal 01], the authors describe a reconstruction algorithm based on the principle of Expectation-Maximization (EM) which accurately and efficiently estimate the shape of the original distribution. In this example, it is also possible to tune the intensity and the type of noise added in order to adjust the level of privacy and the quality of the reconstruction obtained.

The third and last approach is very different in spirit from the first two and attacks the problem with a *cryptography-based view* [Clifton *et al* 02]. After all any distributed and privacy-preserving computation can be considered as an instance of a secure multiparty computation, and privacy-preserving data mining is no exception. In [Lindell and Pinkas 02], the authors described an algorithm which computes a *modified version of ID3* in a distributed and privacy-preserving manner. ID3 is well-established algorithm which uses a *entropy-based metric* in order to construct a decision tree. The version of ID3 that is computed is not the original ID3 algorithm directly but rather an *approximation* of this algorithm. The approximation differs from the exact version by a factor that is a parameter of the algorithm, and has a direct impact on the level of privacy.

The algorithm described in this paper has a cryptographic flavor and belongs to the third approach. In our case, however, the classifier we are after is not a decision tree but rather one which is a result of applying a boosting scheme.

3 Boosting

Boosting is *a method which enables the creation of an efficient classifier by iteratively combining several weak classifiers* whose predictions have to be only slightly better than a random guess. For example, a *weak classifier* (sometimes called a *weak hypothesis*) could be a simple rule which makes a prediction of the class of an object by looking at the value of a single attribute. Boosting can be seen as a *meta-algorithm* because it combines the output of other algorithms.

The idea of boosting originated from the domain of PAC (Probably Approximately Correct) Learning [Valiant 84], which is at the crossing of theoretical computer science and machine learning. However, it has recently seen a resurgence and became popular in the machine learning community with the appearance of the algorithm AdaBoost [Freund and Schapire 97]. Since that time, this algorithm has been studied extensively both theoretically and experimentally, and has been shown to have several interesting properties. Among them, it has been proven that AdaBoost *makes the training error decrease exponentially fast with the number of iterations*, and also that it is able to *make the generalization error continuing to get lower even after the training error has become zero*. We will illustrate the concept of boosting algorithm by briefly reviewing how AdaBoost works.

Formally, a boosting algorithm takes as input a *training set* and a *family of weak classifiers* and return as output a final classifier which is a *linear combination of several weak classifiers*. The training set is a collection of n *data points* $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ where each data point (x_j, y_j) is a pair composed of the *object* itself x_j and its associated *class* y_j. The object x_j is usually represented as *a vector of* k *attributes* where x_{ji} denotes the value of the attribute i for this particular object j. For the rest of the paper, we will limit ourselves to attributes whose values are *real numbers of finite precision*, but there are ways of adapting the algorithm for all kinds of datatypes including *integers*, *binary values* or even *symbolic ones*. For the *binary classification*, the associated class y_j belongs to the set $\{-1, +1\}$ where the label "-1" stands for the *negative class* and "$+1$" for *the positive class*.

One of the key features of a boosting algorithm is the ability to *allocate weights to the training points* which typically represent *how hard the points tend to be to classify*. If the weight of a data point is high, then it means that it is difficult to classify it correctly. On the contrary if its weight is low, then it means that it is relatively easy correctly classify this point. Let w_j denote the weight of the j^{th} data point. The weights are always normalized, that is, $\sum_{j=1}^{n} w_j = 1$. Initially, to indicate that all the data points are equally difficult (or easy) to classify, the weight of the all the data points are set to $\frac{1}{n}$.

Boosting can be seen as an *iterative process that at each iteration tries to find the optimal weak classifier for the current distribution of the weights on the data points*. By optimal, we mean a weak classifier that minimizes *the weighted error of the current distribution*. Let h^t be the weak classifier of the t^{th} iteration and w_j^t the weight of the j^{th} point at the t^{th} iteration. Minimizing the weighted error at the t^{th} iteration simply means finding the weak classifier h^t such that $\epsilon^t = \sum_{j=1}^{n} w_j^t I_{\{h^t(x_j) \neq y_j\}}$ is minimal, where ϵ^t is the weighted error of the t^{th} iteration and $I_{h^t(x_\}j) \neq y_j\}}$ is the indicative function which is 1 if its argument is true, and 0 otherwise.

The weight of h^t, called α^t, indicates how good are the predictions of this classifier and is directly linked to ϵ^t by the formula $\frac{1}{2} \times \ln \frac{1-\epsilon^t}{\epsilon^t}$. After each iteration, the weights of the data points will be updated. If a data point was correctly classified then its weight will decrease, and if it was misclassified its weight will increase. The magnitude by which points will change directly depends on the value of ϵ^t. After T iterations, AdaBoost will output a final classifier which has the form $f^T(.) = \sum_{t=1}^{T} \alpha^t h^t(.)$.

There exist numerous versions and extensions of AdaBoost. In this paper we will use a variant proposed by [Schapire and Singer 99] which allows the classifier not only to answer "-1" or "$+1$", but also to abstain by returning "0". This variant is a particular instance of a more general algorithm which allows the weak classifier to output a real number from $[-1, +1]$. The absolute value of the output can be interpreted as the confidence that the classifier has in its prediction. If the classifier returns "0", this naturally means that it abstains.

4 MABoost

In general, boosting-like algorithms implicitly assume the whole set of data to be available in the hands of one person at the beginning. In our case, we make the assumption that the *dataset is split equally between the two participants* and we would like to design an algorithm that is able to perform boosting while protecting the privacy of the data of the participants at the same time. Therefore our goal is to imagine a distributed and privacy-preserving boosting algorithm. MABoost (Multiparty Abstention Boost), which is described in the following sections, falls in this category. Designing a boosting algorithm that can abstain on some data was not the motivation at the beginning, but it naturally fol-

lows from the necessity to have a distributed and privacy-preserving algorithm. Note that this does not mean that *every* distributed and privacy-preserving boosting algorithm has to use abstention.

The weak classifiers that we use are *decision stumps*. A decision stump is *a very simple decision tree with one root and two leaves*, for example it could be "if attribute i of $x <$ threshold then x belongs to class C1 else x belongs to class C2". This family of classifiers may seem very simple *a priori* but it has been empirically proved that one could obtain excellent results by using them as weak classifiers with a boosting algorithm.

MABoost is an iterative algorithm where each iteration consists of five steps.

Step 1: Finding the ideal decision stump for each attribute

Before trying to agree on a common attribute i, Alice and Bob first computes independently from their respective databases the ideal decision stump for each existing attribute. Let $h_A^{(i)}$ be the weak classifier of Alice for the attribute i and let $\epsilon_A^{(i)}$ be the weighted error of this classifier on her database. $h_B^{(i)}$ and $\epsilon_B^{(i)}$ are defined in the same manner for Bob.

Step 2: Agreeing on a common attribute

Alice and Bob are now searching for an attribute i such that $\epsilon_A^{(i)} < \frac{1}{2}$ and $\epsilon_B^{(i)} < \frac{1}{2}$ simultaneously. If such an attribute does not exist, they want to be aware of that and if there are one or more attributes that meet this requirement they want to find o*ne determined at random among all possible ones*.

To make it easier to apprehend, we will rephrase this problem into a totally equivalent well-studied problem. Suppose Alice and Bob each have a agenda with a list of k time slots. Associated with each time slot is a binary value 0 or 1 indicating if this time slot is already filled or if it is still free. If there are one or more time slots that are free for both, they want to find one chosen at random among all the possibilities, and if there is no time slot where they are both free, they want to detect this situation. The problem of Alice and Bob is that they are not good friends enough to trust each other to the point where one could simply send his entire agenda to the other. Even worse, they do not want to disclose any unnecessary information about their agenda while they try to solve this problem. This problem calls for a solution based on a secure multiparty computation where the function $f(x, y)$ that they want to compute takes x (the agenda of Alice) and y (the agenda of Bob) as input and return the index i of a time slot if a common free time slot exists or "you will never be able to meet each other" if there is no such time slot. We call this task the "*random rendez-vous problem*". In communication complexity, this problem was proposed and analyzed in [Kalyanasundaran and Schnitger 87] and it was proven that it requires a communication of bits of $\Theta(n)$ in the average case. In this last paper, however, there was no interest in any confidentiality or security aspect. It is important to note that *any distributed and privacy-preserving protocol has to use at least as many bits of communication as the best non privacy-preserving distributed protocol*, therefore this result implies that any privacy-preserving protocol for this task will have a communication complexity of at least $\Omega(n)$ bits, but possibly more, because security usually comes with a price.

Alice and Bob will encode their agendas as the bit strings x and y, respectively, where x and $y \in \{0, 1\}^k$, and k is the number of time slots in their agenda. For $1 \leq i \leq k$, x_i equals 1 if Alice is free during the i^{th} time slot and equals 0 if she has already scheduled something for that time. y_i is defined in the same manner for Bob. Suppose that Alice and Bob are given the protocol ORBAN (for the OR of all Bitwise ANd) [Yao 86] which can securely compute the function $\bigvee_{i=1}^{k}(x_i \wedge y_i)$ where $\bigvee$ denotes the usual OR which is true if *one or more* of its arguments are true and $\wedge$ denotes the AND function which returns true *only* if both of its arguments are true. Semantically, we will consider the value "1" to

be equivalent to "true" and the value "0" to represent the value "false". ORBAN(x,y) will return 1 if Alice and Bob have a common time slot and 0 otherwise.

Consider now the following random rendez-vous protocol which uses ORBAN as a subroutine:

RandomRendezVous(x,y)
$v \leftarrow$ ORBAN($(x_1, ..., x_k), (y_1, ..., y_k)$)
If v = 0 then
    Return "you will never be able to meet each other"
Else
    Permute x and y using the same random permutation
    $s \leftarrow 1$
    $p \leftarrow k \div 2$
    While $p \geq 1$ do
        $v \leftarrow$ ORBAN($(x_s, ..., x_{s+p-1}), (y_s, ..., y_{s+p-1})$)
        If v = 0 then
            $s \leftarrow s + p$
        $p \leftarrow p \div 2$
    Return s

We can assume without loss of generality that the total number of time slots k is a power of two. If it is not the case, it is always possible to add some imaginary time slots which are already filled in order to reach this condition. First, Alice and Bob will try to use ORBAN with their whole agendas and they will know if the function returns 0 that they have no empty time slot in common. On the contrary, if this first call to ORBAN returns 1 then this means that there is at least one common time slot which is free for both. Once they have this confirmation, they can search for this time slot by applying a procedure similar to binary search and which tries to locate the empty common slot by putting aside half of the search space at each iteration. Therefore in $\log k$ iterations, they find a common empty time slot. For this time slot to be chosen at random among all the possible free time slots, Alice and Bob *randomize* their inputs by both applying the same random permutation on the indices of their time slots. By doing so, they can avoid the situation where they always find the common free time slot whose index is the lowest, and instead they will find a free time slot randomly distributed among all the possible ones.

Imagine now that we use the random rendez-vous protocol not with an agenda, but rather with a list where a slot is set to 1 if the participant was able to find a weak classifier with weighted error below the threshold $\frac{1}{2}$ for the attribute associated to this slot, and to 0 otherwise. If Alice and Bob each make such a list and then apply the random rendez-vous protocol, they will be able to discover a common attribute where they each have a weak classifier with error bounded by $\frac{1}{2}$ if such an attribute exists, and otherwise they will know that such an attribute does not exist. In the latter case, they will stop the execution of MABoost and return as output the final classifier f^{t-1}.

Step 3: Combining two weak classifiers into a weak classifier that can abstain

Once Alice and Bob have agreed on an attribute i, they will combine the two weak classifiers they each computed for this attribute and that only described their own data into a classifier that will cover the whole set of data and whose weighted error is strictly bound from above by $\frac{1}{2}$. Let h_A be the weak classifier of Alice which is in the form "If attribute's value $< \theta_A$ then object belongs to class C1 else object belongs to class C2", where C1 and C2 represent respectively the positive class and the negative class, or *vice et versa*. We call ϵ_A the weighted error of this classifier. For Bob, we define h_B, his weak classifier, and ϵ_B its associated weighted error in the same manner. Note that in the case of the binary classification, h_B can take two forms; either it follows the same form as Alice classifier,

meaning it can be written as "If attribute's value $< \theta_B$ then object belongs to class C1 else object belongs to class C2", or the classes C1 and C2 are reversed in the rule's description. It is important to notice that in the first case, h_A and h_B both agree on their prediction if the value of the attribute is below the *minimum*(θ_A, θ_B) OR above the *maximum*(θ_A, θ_B), whereas in the second case their classifiers agree if the value of the attribute is above the *minimum*(θ_A, θ_B) AND below the *maximum*(θ_A, θ_B). It is this very simple but important observation that makes it possible to combine efficiently the two classifiers together.

Suppose without loss of generality than $\theta_A < \theta_B$ and that h_A is the rule "If attribute's value $< \theta_A$ then object belongs to class C1 else object belongs to class C2". In the first case (if the decision stumps of Alice and Bob agree in the classes C1 and C2 described by the clauses "then" and "else"), then the two classifiers h_A and h_B are merged into one classifier $\hat{h}$ described by the following rule:

If attribute's value $< \theta_A$
Then object belongs to class C1
Else if attribute's value $> \theta_B$
Then object belongs to class C2
Else abstain

If on the other hand, h_B is in the opposite case and describe by a rule which has the form 'If attribute's value $< \theta_B$ then object belongs to class C2 else object belongs to class C1", then the combined classifier $\hat{h}$ will look like:

If $\theta_A <$ attribute's value $< \theta_B$
Then object belongs to class C2
Else abstain

In both cases, we will obtain a classifier $\hat{h}$ which returns the result of h_A and h_B when they both agree and abstains otherwise. The weighted error of this classifier will be strictly below $\frac{1}{2}$. It is now possible to apply boosting with this classifier $\hat{h}$ which covers the whole space of data.

Step 4: Computing the weight of the merged classifier

If we assign arbitrarily the labels "-1" and "$+1$" to the classes C1 and C2, and the label "0" to an abstention, then $\hat{h}$ returns a value from the set $\{-1, 0, +1\}$. This situation of boosting by abstention is studied in details in [Schapire and Singer 99], and in particular the rules for computing the weight of the merged classifier and how to update the weights of the data points are described there. Let ϵ_- be the weighted error of the classifier $\hat{h}$, ϵ_0 its weighted abstention rate and ϵ_+ the weighted rate of good answers. Alice and Bob can easily determine these three values by applying $\hat{h}$ on their own databases and then putting together their results. We called ϵ_{A-} the weighted error of Alice obtained by using $\hat{h}$ on her database, ϵ_{A0} her weighted abstention rate and ϵ_{A+} the weighted rate of good predictions. ϵ_{B-}, ϵ_{B0} and ϵ_{B+} are defined similarly for Bob. Remember that we assume that Alice and Bob each has half of the whole dataset, therefore we have the simple relations that $\epsilon_- = \epsilon_{A-} + \epsilon_{B-}$, $\epsilon_0 = \epsilon_{A0} + \epsilon_{B0}$ and $\epsilon_+ = 1 - \epsilon_0 - \epsilon_-$. The weight α of the merged classifier only depends on ϵ_- and ϵ_+ and is equal to $\frac{1}{2} \times \log \frac{\epsilon_+}{\epsilon_-}$.

In order to smooth the predictions and to avoid the case where α could become extremely large or even infinite in magnitude when ϵ_- is very small or equal to zero, it has been proposed to pose α has being equal to $\frac{1}{2} \times \log \frac{\epsilon_+ + \delta}{\epsilon_- + \delta}$, where δ is a small appropriate constant.

Step 5: Updating the weight of the data points

Updating the weights of the data points can be made *independently* by Alice and Bob once they know the description of the merged classifier $\hat{h}$ and the weight α assigned to this classifier. All the points where $\hat{h}$ abstains will keep the same weights, all the well-classified points will see their weights decrease, and all the misclassified points will see their weights increase. More precisely, the weight of the well-classified points will be multiplied by $\exp(-\alpha)$ and those of the misclassified points by $\exp(\alpha)$. Finally, the weights are normalized so that $\sum_{j=1}^{n} w_j = 1$.

Once the step 5 is finished, Alice and Bob can choose to start a new iteration or to stop the algorithm and to ouput f^t. The number of iterations t of MABoost they choose to perform can be decided in advance or chosen adaptively depending on some criteria such has the performance of the current classifier they have. The number of iterations also influences the description length of the final classifier, and therefore it is possible to play with this parameter in order to tune the level of privacy.

5 Complexity

When looking at the complexity of a distributed protocol, one can be interested in two different measures of complexity: the *communication complexity*, meaning the number of bits exchanged during the realization of the protocol, and the more classical *algorithmic complexity* which looks at the processing time required from the participants.

5.1 Communication complexity

To define the quantity of communication used in one iteration of MABoost means to *quantify the number of bits exchanged* during this iteration. First, it is easy to observe that the only steps which require some communication between Alice and Bob are the second, the third and the forth. During the first step, Alice and Bob compute independently their ideal weak classifiers for each attribute and therefore they do not need to speak to each other. The same situation occurs during the fifth step where they do not need to engage in a dialog in order to reweight their data points. In fact, the only step that costs in terms of communication is the second one where they have to agree on a common attribute. Let k be the number of attributes of an object. Alice and Bob will use the random rendez-vous protocol to find the common attribute upon which they build the $\hat{h}$ classifier. This protocol requires $\log k$ calls to the protocol ORBAN in the worst case and one call to ORBAN at best if they are able to discover directly that they will not be able to agree on a common attribute. If this case occurs then they stop MABoost and return directly f^{t-1}. Note that this situation can be considered marginal as it happens at most once. The average complexity of the random rendez-vous protocol is the same as the worst case complexity, and is in the order of $\Theta(complexity\ of\ ORBAN\ protocol \times \log k)$. There exists more than one ways of implementing the ORBAN protocol, but if we use in a straightforward manner the technique described in , this complexity will be of $\Theta(n)$ where *n* is the length of the bit strings entered as inputs to the protocol. The overall complexity of the random rendez-vous protocol will therefore be $\Theta(k \log k)$. During the third and forth steps, some communication will also occur, and the cost of this communication is $\Theta(1)$ because the exchange of the descriptions of h_A and h_B necessary for the creation of $\hat{h}$ and communicating ϵ_{A-}, ϵ_{A0}, ϵ_{B-} and ϵ_{B0} can be done with a number of bits that is constant for a fixed finite precision. Therefore the cost of steps 3 and 4 is negligible compared to step 2, and the dominance of the latter make the overall cost of one iteration of MABoost being equal to $\Theta(k \log k)$. The cost of the whole protocol will be $\Theta(tk \log k)$, where *t* is the total number of iterations of MABoost.

5.2 Algorithmic complexity

The only moment where Alice and Bob require some work from their respective computers is at the beginning of the iteration, during the first step when they both have to compute the best ideal weak classifiers for each attribute. The naïve way to find the best weak classifier for an attribute is to first sort the objects according to this attribute and then to find the optimal threshold by looking at the objects in order to find the threshold Θ at which the weighted error is minimized. What is more costly is to sort the objects, which can be done in $\Theta(n \log n)$ if we are only concerned by one attribute, the rest of the operations being negligible compared to this cost. As the sorting has to be repeated for each attribute, the global cost for the k attributes will be in $\Theta(kn \log n)$ for the first iteration. Note that is enough to sort once the objects for each attribute during the first iteration as it is possible to store the results in order to use them for the remaining iterations. The storage of these values has a space complexity of $\Theta(kn)$. Computing the error of the $\hat{h}$ classifier and reweighing the points can be done in $\Theta(n)$ as it only requires looking at each individual data point once in order to perform both tasks. Therefore we have a complexity of $\Theta(kn \log n)$ for the first iteration because of the sorting process and a complexity of $\Theta(kn)$ for the remaining iterations. The overall time complexity of MABoost is a function of t the number of iterations and is in the order of $\Theta((t + \log n)kn)$. Notice that this complexity increases linearly with the number of attributes and so if this number is big, then the processing time of MABoost may become large, also.

6 Security of the protocol

A multiparty protocol can be considered totally secure if the execution of this protocol does not reveal more information than the output of the protocol itself. In cryptography, this is called the *simulation paradigm* [Goldreich 04]. Intuitively, this means that if the view of the protocol of a participant can be efficiently simulated solely based on his input and the output of the protocol, then the different participants learn nothing from the execution of the protocol other than its output. A different definition of security exists which is based the comparison of the output of a real protocol and that of an ideal protocol where an incorruptible third party computes the function in place of the participants and then returns the result to them. In this definition, the main idea is to find a real protocol which emulates the behaviour of the ideal protocol. In this paper, however, we will not be interested in this definition as it is equivalent to the first one for the setting of semi-honest participants which is the one considered in this paper.

Imagine a simulator that has access to the input of Alice, her database, and to the output of the protocol, the final classifier. Would it be possible for this simulator to create a transcript of the execution of the protocol which would be indistinguishable from a real transcript of the execution of the protocol? If we can answer positively to this question, then this means we can assert that the protocol privately computes f, in particular, in the sense of the first definition given in the previous paragraph. For the analysis of the security of the protocol, we only have to concentrate on the second, third and fourth steps as they are the only steps where communication takes place. It is important to observe that in the case of MABoost the process of creating the final classifier (the output of the protocol), is iterative and that the information exchanged in the clear between Alice and Bob during steps 3 and 4 is explicitly contained inside the description of this final classifier. This observation gives us for "free" the security aspect of these two steps. Indeed, why care to protect the privacy of some information during the execution of the protocol when this information is going to be revealed at the end of the protocol anyway. Therefore, the only step we have to be careful about for the analysis of the security of the protocol is the second step.

The second step consists of the random rendez-vous protocol which heavily relies on the

ORBAN protocol as a subroutine. For this step to be considered secure, Alice and Bob must learn nothing other than the index i of a common attribute where they both have a weak classifier with weighted error lower than $\frac{1}{2}$. Suppose that the ORBAN protocol is secure. The first call to ORBAN will only reveal to Alice and Bob whether or not they can agree on a common attribute. If it is not the case, then they stop the protocol and they have learned no extra information. On the contrary if Alice and Bob are able to pursue the protocol, they select half of the attributes and make another call to ORBAN. From this call they will learn whether or not there is at least one attribute they can agree upon in this half of the attribute space. If ORBAN returns 1, they learn that there is at least one potential attribute in the part of the attribute space but nothing else about the other half of the space. On the contrary, if ORBAN returns 0, this means that they know that there is no attribute they can agree upon on the considered part of the attribute space *and* that there is at least one such attribute in the other part of the space. In this case, the protocol has leaked at little bit of extra information. As there will be $\log k$ calls to ORBAN at the maximum, the protocol will reveal $\log k$ bits of extra information before the two agree on a common attribute. This cannot be considered a totally privacy-preserving protocol, and though small we acknowledge that some information is leaked. To which extend someone could do something useful with this type of information is still an open question. Note that it may possible to have a total security by using more complicated and sophisticated protocols which solve the set intersection problem [Freedman *et al* 04] but this would come with an increase of the communication cost.

7 Possible generalizations and conclusion

In this paper, we have presented a protocol which performs a boosting-like algorithm in a distributed and privacy-preserving manner in the restricted case where we only deal with two participants and where the weak classifiers have the form of decision stumps. There is at least two natural ways of generalizing this protocol, first by allowing the number of participants to be greater than two, and also by looking at other families of weak classifiers such as for example more complex decision trees or neural networks with few units in the hidden layer.

To extend this protocol to the case of *n* participants, with $n \geq 2$, we could for example define at each iteration a classifier $\hat{h}$ which carries out a majority vote from the individual weak classifiers h_1, h_2, ..., h_n of the different participants. The classifier could for example return the vote of the majority of the classifiers when this majority is above a threshold set to $\frac{1}{2} + \delta$ and abstain otherwise, δ being a well-chosen constant between 0 and $\frac{1}{2}$ whose value can be tuned in order to increase the confidence of the classifier $\hat{h}$ in its prediction. Notice that the protocol described in this paper is a particular instance of this more general method.

It could be interesting also to test this algorithm with other types of weak classifiers and to compare the results. Particularly, it would be interesting to observe empirically how the choice of the type of weak classifiers, the number of bits required to describe these classifiers (which in a sense is a measure of their complexity), and the number of iterations influence the convergence speed of MABoost. For example, it may be conceivable that with richer family of weak classifiers such as decision trees generated by the algorithm C4.5, one can obtain an identical or better performance than with the decision stumps in fewer iterations, meaning that the convergence would be faster. However this would come at the price of exchanging more bits of communication at each iteration because the decision trees inferred by C4.5 require a greater (and possibly unbounded) number of bits to be described than the decision which only requires a constant number because of their inherent simplicity. One could define a criteria to take into account at the same time the number of iterations and the expected size needed to describe the weak classifiers,

which would be able to compare two possible choices of family of weak classifiers even in case of equal performance. Intuitively, we are searching for weak classifiers that can be described with few bits but which at the same time, yield a good performance after a "sensible" number of iterations of MABoost. This would make it possible to bound directly the number of bits exchanged during the protocol and therefore will give an upper bound on the quantity of information which has been communicated.

For some families of weak classifiers, there must exist ways of combining weak classifiers without requiring the resulting classifier $\hat{h}$ to abstain on a part of the dataset. Being able to combine two or more weak classifiers into a weak classifier that do not abstain and which answer "-1" or "$+1$", permits to directly apply the original AdaBoost in a distributed manner without using an abstention variant of boosting. Potentially the convergence could be a little bit faster in practice in this situation.

The work we have described in this paper is still in progress right now and among several experimentations, we are planning to compare empirically on several datasets the original AdaBoost algorithm with MABoost in terms of performance and convergence speed. This will help us evaluate how much the privacy-preserving aspect has an influence on the performance of the final classifier. We are also currently investigating the privacy-preserving part of step 2, and in particular we would like to design a efficient randomized version of the set intersection problem.

References

[Agrawal and Ramakrishnan 00] R. Agrawal and R. Ramakrishnan, "Privacy preserving data mining", *Proceedings of the ACM SIGMOD on Management of Data*, pp. 439 – 450, 2000.

[Agrawal and Aggarwal 01] D. Agrawal and C.C. Aggarwal, "On the design and quantification of privacy preserving data mining algorithms", *Proceedings of the 20th ACM Symposium of Principles of Databases Systems*, pp. 247 – 255, 2001.

[Atallah *et al* 99] M.J. Atallah, E. Bertino, A.K. Elmagarmid, M. Ibrahim and V.S. Verykios, "Disclosure limitations of sensitive rules", *Proceedings of the IEEE Knowledge and Data Engineering Workshop*, pp. 45 – 52, 1999.

[Chaum *et al* 88] D. Chaum, C. Crépeau and I. Damgard, "Multiparty unconditionally secure protocols", *Proceedings of the 20th Annual Symposium on the Theory of Computing*, pp. 11 – 19, 1988.

[Clifton *et al* 02] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin and M.Y. Zhiu, "Tools for privacy preserving distributed data mining", *SIGKDD Explorations* **4**(2), pp. 28 – 34, 2002.

[Evfimievski *et al* 03] E. Evfimievski, J.E. Gehrke and R. Srikant, "Limiting privacy breaches in privacy preserving data mining", *Proceedings of the 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Databases Systems*, 2003.

[Freedman *et al* 04] M. Freedman, K. Nissim and B. Pinkas, "Efficient private matching and set intersection", *Proceedings of Eurocrypt'04*, pp. 1 – 19, 2004.

[Freund and Schapire 97] Y. Freund and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting", *Journal of Computer and System Sciences* **55**(1), pp. 119 – 139, 1997.

[Goldreich 04] O. Goldreich, *Foundations of Cryptography: Volume II Basic Applications*, Cambridge University Press, 2004.

[Kalyanasundaran and Schnitger 87] B. Kalyanasundaran and G. Schnitger, "The probabilistic communication of set intersection", *Proceedings of the 2nd Annual IEEE Conference on Structure in Complexity Theory*, pp. 41 – 47, 1987.

[Lindell and Pinkas 02] Y. Lindell and B. Pinkas, "Privacy preserving data mining", *Journal of Cryptology***15**, pp. 177 – 206, 2002.

[Schapire and Singer 99] R. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions", *Machine Learning* **37**(7), pp. 297 – 336, 1999.

[Valiant 84] L. Valiant, "A theory of the learnable", *Communications of the ACM* **27**(11), pp. 1134 – 1142, 1984.

[Verykios *et al* 04] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin and M.Y. Zhiu, "State-of-the-art in privacy preserving data mining", *SIGMOD Record* **33**(1), pp. 50 – 57, 2004.

[Yao 86] A.C. Yao, "How to generate and exchange secrets", *Proceedings of the 27th Symposium on Foundations of Computer Science* **27**(11), pp. 162 – 167, 1986.